Nonlinear Inelastic Finite Element Analysis of Reinforced Concrete Structures With Emphasis on Shear and Torsion. (Volumes I and II).

Ananth Ramaswamy
Louisiana State University and Agricultural & Mechanical College

Follow this and additional works at: https://digitalcommons.lsu.edu/gradschool_disstheses

Recommended Citation

This Dissertation is brought to you for free and open access by the Graduate School at LSU Digital Commons. It has been accepted for inclusion in LSU Historical Dissertations and Theses by an authorized administrator of LSU Digital Commons. For more information, please contact gradetd@lsu.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.
Nonlinear inelastic finite element analysis of reinforced concrete structures with emphasis on shear and torsion. (Volumes I and II)

Ramaswamy, Ananth, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1992

Copyright ©1993 by Ramaswamy, Ananth. All rights reserved.
NONLINEAR INELASTIC FINITE ELEMENT ANALYSIS OF REINFORCED CONCRETE STRUCTURES WITH EMPHASIS ON SHEAR AND TORSION Volume I

A Dissertation

Submitted to the Graduate Faculty of the Louisiana State University and Agricultural and Mechanical College in partial fulfillment of the requirements for the degree of Doctor of Philosophy

in

The Department of Civil Engineering

by

Ananth Ramaswamy
B.Tech., Indian Institute of Technology, 1985
M.S. University of California, 1986
May, 1992
Acknowledgements

This study was conducted under the supervision of Dr.F. Barzegar, formerly Assistant Professor of Civil Engineering, LSU. The work was completed under the supervision of Dr.G.Z. Voyiadjis, Professor of Civil Engineering, LSU. I wish to thank both of them for their guidance, encouragement, and the many useful discussions I had with them while conducting this study.

I would also like to thank the members of my doctoral advisory committee - Professor R.R. Avent, Professor V.K. Gopu, Professor S.S. Iyengar, Professor P.N. Kirk, and Dr.P. Zuraski for reviewing this dissertation and offering their helpful suggestions.

Financial support provided by the Department of Civil Engineering, LSU and the National Science Foundation while conducting this study is most gratefully acknowledged. A special word of thanks to the Faculty and staff, Department of Civil Engineering, LSU, for providing computational hardware peripherals while conducting this study.

This study was conducted using the IBM 3090/MVS mainframe computer. The funds provided by the university for this purpose is gratefully acknowledged. A special word of thanks to Prof. W.F. Beyer, Director of SNCC, LSU, for his cooperation and encouragement. The help provided by the SNCC staff, in particular, Dr.M. Foroozesh, Mr.P.J. Quinn, and Ms.D.D. Sutton is also gratefully acknowledged.

I wish to thank Dr.C. Channakeshava, Research Associate at LSU, for the many useful discussions I have had with him, during the course of this study. I also wish to thank my colleagues - Mr.S. Maddipudi, Mr.E. Rainer, Mr.A. Venson, Mr.A. Taher, Mr P. Kurup, Dr.H. Pentas, Mr.A. Puppala and Dr.K. Rebello; who have made my graduate school experience at LSU a pleasant one. Mr.A. Venson's knowledge and help in using 'LATEX' deserves special mention. Mr.E. Rainer's help in processing slides for the presentation of this work is also gratefully acknowledged.
A special word of thanks to my friends and roommates during the course of this study - Mr.S. Joykutty, Mr.S. Natarajan, Dr.J. Rajan, Dr.S. Rajanaryanan, Mr.K. Rangan, Mr.Y.S. Rao, Dr.K. Umesh, Mr. A. Vibhas, Mr. and Mrs.R. Seval, Mr.S. Shivakumar, Mr and Mrs.G. Thiagarajan, Dr.L. Wargo and many others for making life away from school a wonderful experience.

Words cannot express my heartfelt thanks to my parents for their constant encouragement, moral support and financial assistance throughout the course of my education. I also wish to thank my uncle Mr. Sampath Iyengar for his general interest in my progress. My brother Murali, and my sister-in-law Vinita also deserve a special word of thanks for their constant encouragement through the course of my studies. My newly born niece Tara also deserves mention: her cheerfully smiling innocent face has brought added cheer to my day.
Contents

Volume I

Acknowledgements ii

Table of Contents iv

List of Tables vii

List of Figures viii

List of Variables xiii

Abstract xv

1 Introduction 1
  1.1 Objectives, Scope and Limitations ......................... 5
  1.2 Organization .................................................. 6

2 Literature Review 8
  2.1 Introduction .................................................. 8
  2.2 Constitutive Modelling of Steel ............................ 8
  2.3 Constitutive Modelling of Uncracked Concrete .......... 9
    2.3.1 Failure Criteria for Concrete ....................... 9
    2.3.2 Stress-Strain Response in Concrete ................. 11
      2.3.2.1 Elasticity Based Models .......................... 11
      2.3.2.2 Plasticity Based Models ......................... 14
      2.3.2.3 Plastic Fracturing Theory ....................... 17
      2.3.2.4 Endochronic Theory Based Models ............... 18
      2.3.2.5 Damage Mechanics Based Models ................. 19
      2.3.2.6 Comparative Studies of Constitutive Models .... 20
  2.4 Crack Simulation and Post-Cracking Nonlinearities .... 21
    2.4.1 Crack Simulation ....................................... 21
    2.4.2 Post-Cracking Nonlinearities ....................... 24
      2.4.2.1 Strain-Softening .................................. 24
      2.4.2.2 Tension-Stiffening ............................... 25
      2.4.2.3 Bond-Slip ........................................ 27
      2.4.2.4 Aggregate Interlock and Dowel Action .......... 28
      2.4.2.5 Compression Softening of Cracked Concrete .... 28
List of Tables

6.1 Dimensions and Material Properties of Uniaxial Tension Specimen 102
6.2 Results of Uniaxial Tension Tests on RC Specimen 102
6.3 Loading Procedures for Tests on Membrane Elements 108
6.4 Dimensions and Material Properties of Membrane Elements 109
6.5 Results of Tests on Membrane Elements 113
6.6 Loading Procedures For Tests on Membrane Elements 123
6.7 Dimensions and Material Properties of Membrane Elements 124
6.8 Results of Tests on Membrane Elements 124
6.9 Dimensions of Shearwall Test Specimens 142
6.10 Material Properties of Shearwall Specimens 142
6.11 Results of Tests on Shearwalls 143
6.13 Material Property of RC Slab Element Subjected to General Loadings 188
List of Figures

1.1 Reinforced Concrete Structural Components ............................................ 2

2.1 Representation of Steel in Finite Element Analysis of Reinforced Concrete (a) Smeared Steel Layer (b) Embedded Steel Element (c) Discrete Bar Element .......................................................... 10

2.2 Representation of Cracks in Finite Element Analysis of Concrete (a) and (b) Discrete Cracks (c) Smeared Cracks ........................................... 22

2.3 Stress Distribution in Cracked Reinforced Concrete .................................. 26

3.1 Failure Envelope for Concrete Based on Stresses ..................................... 32

3.2 Comparison of Ottosen Strength Envelope with (a) Triaxial and (b) Biaxial Test Data, Ottosen (1977). .............................................................. 34

3.3 Unloading and Reloading Response of Concrete Along a Secant Path. 36

3.4 Determination of the Nonlinearity Index $\gamma$. ............................................ 37

3.5 Computation of Secant Stiffness, $E_{se}$ in the Post-Peak Region. .................. 39

3.6 Computation of Poisson’s Ratio, $\nu_{iso}$ for Concrete. ................................. 41

3.7 Comparison with Uniaxial Compressive Loading Test Data. ......................... 42

3.8 Triaxial Loading Along the Compressive Meridian. ....................................... 43

3.9 Triaxial Loading Along the Compressive Meridian. ....................................... 43

3.10 Triaxial Loading Along the Tensile Meridian. ............................................ 44

3.11 Comparison with Triaxial Non-proportional Loading Data. ......................... 45

3.12 Stress-Strain Response of Steel ................................................................. 46

4.1 Plain Concrete Element at Incipient Cracking ............................................ 48

4.2 (a) Cracked Plain Concrete Specimen, (b) Rheological Representation of Plain Concrete Cracking, (c) Stress-Strain Relationship For Cracked Plain Concrete, (d) Stress-Strain Relationship for Uncracked Concrete, (e) Stress-Strain Relationship for Cracks. ............................................. 50

4.3 (a) RC Element, (b) Rheological Representation of Reinforced Concrete Cracking, Rots (1988) ................................................................. 52

4.4 Rheological Representation of Reinforced Concrete Cracking, Present Study ................................................................. 54

4.5 Tri-Directionally Reinforced Concrete Element Subjected to General Loads Prior to Cracking ................................................................. 57

4.6 (a) Rheological Representation of a Tri-Directionally Reinforced Concrete Element Prior to Cracking (b) Schematic Representation of the Element. ............................................. 58
4.7 Tri-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Strain $\epsilon < \epsilon_{cr}$ in the Steel Direction. 59
4.8 Strain-Softening Envelope Employed Normal to the Cracks in Cracked RC, With Steel Direction Strains $\epsilon < \epsilon_{cr}$ 61
4.9 Tri-Directionally Reinforced Concrete Element Subjected to General loads, With Atleast One Strain $\epsilon \geq \epsilon_{cr}$ in the Steel Direction. 62
4.10 (a) Rheological Representation of a Tri-Directionally Reinforced Concrete After Full Crack Opening (b) Schematic Representation of a Cracked Point. 63
4.11 Bi-Directionally Reinforced Concrete Element Subjected to General Loads Prior to Cracking. 65
4.12 Bi-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Strain $\epsilon < \epsilon_{cr}$ in the Steel Direction. 66
4.13 Bi-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Atleast One Strain $\epsilon \geq \epsilon_{cr}$ in the Steel Direction. 67
4.14 Uni-Directionally Reinforced Concrete Element Subjected to General Loads Prior to Cracking. 69
4.15 Uni-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Strain $\epsilon < \epsilon_{cr}$ in the Steel Direction. 70
4.16 Uni-Directionally Reinforced Concrete Element Subjected to General Loads, With Strain $\epsilon \geq \epsilon_{cr}$ in the Steel Direction. 71
4.17 Compression-Softening in Cracked Reinforced Concrete, Vecchio and Collins (1982) 73
4.18 Compression-Softening in Cracked Reinforced Concrete, Present Study 75
4.19 Experimental Setup Used to Study Crack Interface Shear Load vs. Displacement Behavior Under Lateral Confinement ($\sigma_c$), Tassios and Vintzeleou (1987). 77
4.20 Crack Interface Shear Stress-Strain Curves at Various Levels of Confining Stresses Due to Aggregate Interlock and Dowel Action, Tassios and Vintzeleou (1987). 78

5.1 Eight-Noded Degenerate Shell Element. 80
5.2 Implicit Layering Procedure 84
5.3 Explicit Layering Procedure 87
5.4 Secant Iterative Procedure. 88
5.5 Convergence Path at a Point in Concrete Prior to Crushing. 90
5.6 Post-Peak Convergence Path in Concrete. 92
5.7 Convergence Path in a Strain Softening Cracked Point. 93
5.8 Detection and Correction of Overshooting at a Cracked Point. 94
5.9 Convergence Path at a Tension-Stiffening Point. 95

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
5.10 Detection and Correction for Overshooting at a Tension-Stiffening Point ................................................. 96
5.11 Convergence Path in Steel in the Post-Yielded Zone ................................................................. 97
5.12 Detection and Correction for Overshooting in Steel ............................................................... 98

6.1 Tensile Stress-Strain Response, Specimen Shima #3 ................................................................. 103
6.2 Axial Load-Deformation Response, Specimen Shima #3 ................................................................. 103
6.3 Tensile Stress-Strain Response, Specimen Shima #5 ................................................................. 104
6.4 Axial Load-Deformation Response, Specimen Shima #5 ................................................................. 104
6.5 Tensile Stress-Strain Response, Specimen Rizkalla #2 ................................................................. 106
6.6 Axial Load-Deformation Response, Specimen Rizkalla #2 ................................................................. 106
6.7 Tensile Stress-Strain Response, Specimen PB13 ................................................................. 107
6.8 Axial Load-Deformation Response, Specimen PB13 ................................................................. 107
6.9 Axial Stress-Strain Response, Specimen S1 ................................................................. 110
6.10 Axial Stress-Strain Response, Specimen S3 ................................................................. 111
6.11 Axial Stress-Strain Response, Specimen S5 ................................................................. 111
6.12 Axial Stress-Strain Response, Specimen S6 ................................................................. 112
6.13 Axial Stress-Strain Response, Specimen S7 ................................................................. 112
6.14 Load vs. Strain Response, Specimen PB21 ................................................................. 115
6.15 Load vs. Transverse Strain Response, Specimen PB21 ................................................................. 115
6.16 Load vs. Longitudinal Strain Response, Specimen PB21 ................................................................. 116
6.17 Load vs. Maximum Principal Strain Response, Specimen PB21 ................................................................. 116
6.18 Tensile Stress-Strain Response, Specimen PB21 ................................................................. 117
6.19 Principal Stress Direction vs. Axial Load, Specimen PB21 ................................................................. 117
6.20 Principal Strain Direction vs. Axial Load, Specimen PB21 ................................................................. 118
6.21 Load vs. Strain Response, Specimen PB18 ................................................................. 120
6.22 Load vs. Transverse Strain Response, Specimen PB18 ................................................................. 120
6.23 Load vs. Longitudinal Strain Response, Specimen PB18 ................................................................. 121
6.24 Load vs. Maximum Principal Strain Response, Specimen PB18 ................................................................. 121
6.25 Tensile Stress-Strain Response, Specimen PB18 ................................................................. 122
6.26 Principal Stress Direction vs. Axial Load, Specimen PB18 ................................................................. 122
6.27 Principal Strain Direction vs. Axial Load, Specimen PB18 ................................................................. 123
6.28 Load vs. Shear Strain Response, Specimen PV19 ................................................................. 125
6.29 Load vs. Longitudinal Strain Response, Specimen PV19 ................................................................. 125
6.30 Load vs. Transverse Strain Response, Specimen PV19 ................................................................. 126
6.31 Tensile Stress-Strain Response, Specimen PV19 ................................................................. 126
6.32 Compressive Stress-Strain Response, Specimen PV19 ................................................................. 127
6.33 Principal Stress Direction vs. Axial Load, Specimen PV19 ................................................................. 127
6.34 Principal Strain Direction vs. Axial Load, Specimen PV19 ................................................................. 128
6.35 Load vs. Shear Strain Response, Specimen PV27 ................................................................. 130
6.36 Load vs. Longitudinal Strain Response, Specimen PV27 ................................................................. 130
6.37 Load vs. Transverse Strain Response, Specimen PV27 ................................................................. 131
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.38</td>
<td>Tensile Stress-Strain Response, Specimen PV27</td>
<td>131</td>
</tr>
<tr>
<td>6.39</td>
<td>Compressive Stress-Strain Response, Specimen PV27</td>
<td>132</td>
</tr>
<tr>
<td>6.40</td>
<td>Load vs. Shear Strain Response, Specimen PV29</td>
<td>133</td>
</tr>
<tr>
<td>6.41</td>
<td>Load vs. Longitudinal Strain Response, Specimen PV29</td>
<td>133</td>
</tr>
<tr>
<td>6.42</td>
<td>Load vs. Transverse Strain Response, Specimen PV29</td>
<td>134</td>
</tr>
<tr>
<td>6.43</td>
<td>Tensile Stress-Strain Response, Specimen PV29</td>
<td>134</td>
</tr>
<tr>
<td>6.44</td>
<td>Compressive Stress-Strain Response, Specimen PV29</td>
<td>135</td>
</tr>
<tr>
<td>6.45</td>
<td>Tensile Stress-Strain Response, Specimen CS6</td>
<td>136</td>
</tr>
<tr>
<td>6.46</td>
<td>Compressive Stress-Strain Response, Specimen CS6</td>
<td>136</td>
</tr>
<tr>
<td>6.47</td>
<td>Dimensions of Shearwall Test Specimen</td>
<td>140</td>
</tr>
<tr>
<td>6.48</td>
<td>Finite Element Mesh For Shearwall Specimen</td>
<td>141</td>
</tr>
<tr>
<td>6.49</td>
<td>Load Displacement Response of Shearwall Specimen SW11</td>
<td>143</td>
</tr>
<tr>
<td>6.50</td>
<td>Load Displacement Response of Shearwall Specimen SW21</td>
<td>144</td>
</tr>
<tr>
<td>6.51</td>
<td>Load Displacement Response of Shearwall Specimen Maier#4</td>
<td>144</td>
</tr>
<tr>
<td>6.52</td>
<td>Load Displacement Response of Shearwall Specimen Maier#9</td>
<td>145</td>
</tr>
<tr>
<td>6.53</td>
<td>Deflection Profile, Specimen SW21</td>
<td>146</td>
</tr>
<tr>
<td>6.54</td>
<td>Compressive Stress Distribution in Shearwall Specimen SW21 Near</td>
<td>147</td>
</tr>
<tr>
<td></td>
<td>Ultimate Load</td>
<td></td>
</tr>
<tr>
<td>6.55</td>
<td>Crack Pattern in Shearwall Specimen SW21 Near Ultimate Load</td>
<td>148</td>
</tr>
<tr>
<td>6.56</td>
<td>Stress Distribution in Horizontal Steel Near Ultimate Load, in Shear-</td>
<td>149</td>
</tr>
<tr>
<td></td>
<td>wall Specimen SW21</td>
<td></td>
</tr>
<tr>
<td>6.57</td>
<td>Stress Distribution in Vertical Steel Near Ultimate Load, in Shearwal</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>Specimen SW21</td>
<td></td>
</tr>
<tr>
<td>6.58</td>
<td>Dimensions of Specimen Delft Beam</td>
<td>153</td>
</tr>
<tr>
<td>6.59</td>
<td>F.E. Mesh and Layering details for Specimen Delft Beam</td>
<td>154</td>
</tr>
<tr>
<td>6.60</td>
<td>Load Vs. Deflection Response, Specimen Delft Beam</td>
<td>155</td>
</tr>
<tr>
<td>6.61</td>
<td>Normal Stress Distribution at Point, x=1340mm, from the Support,</td>
<td>156</td>
</tr>
<tr>
<td></td>
<td>Implicit Layering Procedure</td>
<td></td>
</tr>
<tr>
<td>6.62</td>
<td>Transverse Shear Stress Distribution at a Point, x=1340mm, from the</td>
<td>156</td>
</tr>
<tr>
<td></td>
<td>Support, Implicit Layering Procedure</td>
<td></td>
</tr>
<tr>
<td>6.63</td>
<td>Deflection Profile, Specimen Delft Beam, Implicit Layering.</td>
<td>157</td>
</tr>
<tr>
<td>6.64</td>
<td>Concrete Compressive Stresses-Top Surface, Specimen Delft Beam,</td>
<td>158</td>
</tr>
<tr>
<td></td>
<td>Implicit Layering</td>
<td></td>
</tr>
<tr>
<td>6.65</td>
<td>Reinforcement Stresses, Specimen Delft Beam, Implicit Layering</td>
<td>159</td>
</tr>
<tr>
<td>6.66</td>
<td>Cracking Pattern, Specimen Delft Beam, Implicit Layering.</td>
<td>160</td>
</tr>
<tr>
<td>6.67</td>
<td>Deflection Profile, Specimen Delft Beam, Explicit Layering.</td>
<td>162</td>
</tr>
<tr>
<td>6.68</td>
<td>Concrete Compressive Stresses-Top Surface, Specimen Delft Beam,</td>
<td>163</td>
</tr>
<tr>
<td></td>
<td>Explicit Layering</td>
<td></td>
</tr>
<tr>
<td>6.69</td>
<td>Reinforcement Stresses, Specimen Delft Beam, Explicit Layering.</td>
<td>164</td>
</tr>
<tr>
<td>6.70</td>
<td>Cracking Pattern, Specimen Delft Beam, Explicit Layering.</td>
<td>165</td>
</tr>
<tr>
<td>6.71</td>
<td>Moment vs. Principal Curvature, Specimen ML9</td>
<td>167</td>
</tr>
<tr>
<td>6.72</td>
<td>Dimensions of Specimen Dudeck Slab S1</td>
<td>169</td>
</tr>
</tbody>
</table>
List of Variables

$\sigma_1, \sigma_2, \sigma_3$  Principal Stress Components ($\sigma_1 > \sigma_2 > \sigma_3$)

$\epsilon_1, \epsilon_2, \epsilon_3$  Principal Strain Components ($\epsilon_1 > \epsilon_2 > \epsilon_3$)

$I_1$  First Stress Invariant.

$I_2$  Second Deviatoric Stress Invariant.

$I_3$  Third Deviatoric Stress Invariant.

$f'_c$  Uniaxial Compressive strength (Cylinder).

$\epsilon'_c$  Compressive Strain Corresponding to Uniaxial Compressive Strength

$\sigma_{bc}$  Biaxial Compressive Strength

$\{\sigma\}_{x,y,z}$  Stress Tensor in Global Directions.

$\{\epsilon\}_{x,y,z}$  Strain Tensor in Global Directions.

$[D]^{co}$  Concrete Stiffness Tensor in Global Directions.

$\xi$  Mean Hydrostatic Stress.

$\sigma'$  Deviatoric Stress.

$E_{co}$  Secant Stiffness Modulus of Concrete

$\nu_{co}$  Secant Poisson’s Ratio of Concrete

$\sigma_y$  Yield Stress for Steel.

$\epsilon_y$  Yield Strain for Steel.

$E_y$  Secant Stiffness for Steel.

$[D]^{cr}$  Crack Stiffness Tensor in local Directions.

$E_{cr}, G_{cr1}, and G_{cr2}$  Crack Normal and Shear Stiffness Components.

$G_f$  Fracture energy consumed per unit length of crack.

$[D]^{ts}$  Tension-Stiffening Tensor in the Steel Direction.

$E'^{ts}$  Secant Tension-Stiffening Stiffness Component.

$[D]^{cr-co}$  Plain Cracked Concrete Stiffness Tensor in Global Directions
\[ [D]^{rc-co} \]
Cracked Reinforced Concrete Stiffness Tensor in Global Directions Including Tension-Stiffening.

\[ [D]^{coupling} \]
Coupling Stiffness Tensor in Steel Coordinates.

\[ [D]^{cr-recon} \]
Cracked Reinforced Concrete Including Tension-Stiffening and Coupling.
Abstract

In this study a post-cracking formulation for the analysis of reinforced concrete structures has been proposed. The secant stiffness formulation considers the following nonlinearities: effects of tension-stiffening in concrete along the reinforcing direction(s), aggregate interlock model with variable shear based on crack confining stresses, and reduction of concrete compressive strength and stiffness after cracking as a function of stresses or strains. The cracking model is capable of simulating the effects of multiple non-orthogonal cracks at a point. This total stiffness formulation treats concrete unloading and reloading along a secant path and is capable of simulating the post-peak strain softening response. The reinforcement is treated as an elasto-plastic material with possible strain-hardening having uniaxial stiffness properties. Perfect bond is assumed between concrete and steel.

The proposed nonlinear inelastic secant stiffness model has been implemented into a special purpose finite element program. Implicit and explicit layering procedures have been incorporated in conjunction with shell elements. These layering procedures enable the geometric modelling of different materials through the cross-section. Procedures to compute the variation in the transverse shear stress through the cross section in the implicit layering procedure have been adopted. The transverse shear stress distribution through the cross-section are better represented with these layering procedures and have been included in the material model. The progress of material nonlinearities through the cross section, such as concrete cracking or crushing, are simulated more effectively with these layering procedures.

The capabilities of this ‘three-dimensional’ material model have been explored by simulating a number of experimental test specimens subjected to various loadings and comparing the analytical predictions with appropriate test results and design code provisions. These examples include a number of panels and wall specimen subjected to in-plane loads; beams and slab specimens which are subjected to out-of-plane loads and a slab element representing the connection region of flat plate
column joints subjected to a combination of loads. The results of these studies indicate that the analytical model is capable of capturing the dominant deformational response characteristics and predicting the appropriate failure modes.
Chapter 1

Introduction

Reinforced concrete (RC) surface structures find their use in our daily life as components of buildings: panels, walls, slabs and shell roofs; and in more complex structures such as pressure vessels, storage tanks, etc. (figure 1.1). Due to their planar geometry, surface structures carry most of the load through membrane (in-plane) stresses. The complexities associated with the development of rational analytical procedures for such structures, have caused existing design methods to continue in many respects to be based on empirical procedures, which to a large extent depend on experimental data. The rapid development of computational hardware and powerful numerical procedures such as the finite element method have made feasible the investigations of the load-deformational response of RC surface structures into the inelastic domain.

To understand the behavior of RC structural systems under complex loading configurations, a number of experimental and analytical studies have been conducted in recent years. These studies have focused on components of such systems to examine the response nonlinearities under various loads and to explore their influence on load carrying capacities. Under a general state of stress in RC elements, cracks are formed when the tensile stresses or strains reach their cracking value. After cracking, the tensile stresses in concrete diminish rapidly at the crack surfaces and are carried instead by the reinforcement. However, it is expected that concrete between these cracks continues to carry tensile stresses and thus contribute to the stiffness of the cracked RC element. The concrete between the cracks is also capable of sustaining compressive stresses in a direction parallel to the crack surfaces. In such cases, it is expected that the ultimate strength and stiffness of concrete compressive 'struts' would reduce considerably due to the lack of confining effect, stemming from the opening of cracks. After reaching the peak compressive stress
Figure 1.1: Reinforced Concrete Structural Components.
concrete continues to sustain stresses that reduce gradually with increasing strains. This phenomenon is known as post-peak strain softening. Shear stresses are transferred across open cracks through the interaction of rough interfaces. Because of the presence of shear stresses it is possible for new cracks to form at a different angle and previously existing cracks to close. The opening and closing of cracks, crushing of concrete, yielding of reinforcement, and the slippage of the reinforcement (bond slip) may significantly influence the deformational response and the ultimate capacity of RC elements.

Experimental studies have considered numerous parameters such as: types of structural elements (panels, deep beams, shear walls, slabs), concrete strength, type of reinforcement (undeformed and deformed bars, wire mesh), different loading methods (monotonic, proportional, non-proportional, reverse cyclic, etc.), and various loading distribution (distributed edge loads via steel or concrete, point loads etc.) and boundary conditions (simply supported, clamped etc.). Tests of RC panel elements under the combined actions of membrane shear and normal stresses (Vecchio and Collins 1982; Rizkalla and Hwang 1984; Bhide and Collins 1986; H. Shima 1987) have identified the importance of a number of response parameters including the contribution of the concrete between the cracks in the deformational response of the elements up to failure. The experiments reveal that in some cases the directions of the initial cracks do not remain fixed on continued loading, and that the crack shifting behavior is a function of the anisotropic nature of the problem. Similar conclusions have been reached from flexural and torsional tests on RC slab elements (Cardenas and Sozen 1968; Marti et al. 1987a, 1987b). Tests on heavily reinforced panel and slab elements have revealed that the cracking of concrete significantly reduces the compressive strength of concrete in the orthogonal direction. These experimental observations indicate that in order to develop analytical models to simulate the response of RC structures up to failure, many of the aforementioned response characteristics should be considered.

While RC surface structures are primarily intended and designed to carry the applied loads by means of in-plane (surface) stresses, there are cases where they are also subjected to appreciable out-of-plane shear stresses. Examples include
RC offshore containment vessel subjected to ice loads and connection regions of slab-wall-column systems in monolithic RC structures. In RC flat plate structures the transfer of shear and unbalanced bending moments at slab-column connections is a critical design consideration. This aspect of design becomes very important under the action of lateral loads, due to wind gusts or earthquake excitations, when a substantial bending moment has to be transferred across the connection. In addition to the flexural and torsional moments, resulting from the in-plane stresses, large amounts of out-of-plane shear stress are also mobilized.

Due to the large number of parameters that can influence the behavior of RC structures, experimental investigations can only provide a limited amount of information. Further, interaction between the different parameters cannot be isolated and studied. In most structures a prototype test is not economically feasible and the various size effects make direct extrapolation of results of small scale tests to full scale structures questionable.

The high cost of experimental investigations has necessitated the development of numerical models for the analysis of reinforced concrete structures. These models vary in the degree of complexity both in terms of their theoretical basis and their computer implementation. The limitations in their capabilities stem from inability of the constitutive models for concrete and steel to simulate different response characteristics eg. dilatation of concrete prior to crushing, hardening of steel subsequent to yielding etc. Some models are incapable of simulating other kinds of nonlinearities eg. tension-stiffening, bond slip, aggregate interlock, dowel action etc., properly. In a number of investigations (eg. Darwin and Pecknold 1974; Milford and Schnobrich 1984; Channakeshava 1988; Kolleger 1988), development of the constitutive model for reinforced concrete is based on a plane stress assumption, i.e. neglecting the influence of the out-of-plane shear and normal stress components in material model computations.

The type of numerical solution procedures, e.g. incremental or total formulations; shape and type of finite element used, e.g. triangular or quadrilateral
elements, shell or solid elements; simple linear or higher order displacement elements; the boundary condition idealizations, e.g. simply supported, clamped etc.; can themselves influence the performance of an analytical model.

The primary aim of the present study is to develop a suitable numerical model, based on a three-dimensional formulation, capable of predicting both the deformational response and the ultimate strength of RC surface structures such as panel elements, shearwalls, slabs etc. For this purpose a special purpose finite element three dimensional finite element program having degenerate shell elements (Ahmad et al. 1970) has been developed. To simulate the out of plane shear stress variations, layering procedures (Figueras and Owen, 1983; Barzegar 1989) has been implemented. In this study, emphasis has been placed on simulating the dominant deformational characteristics and identifying governing modes of failure. The objectives, limitations and scope of the present study are presented in the following section.

1.1 Objectives, Scope and Limitations

- Develop a three-dimensional material model for reinforced concrete considering the following: the behavior of concrete under triaxial stress states, crushing and post-crushing behavior of concrete, cracking and post-cracking behavior of plain concrete, tension-stiffening applied to concrete in the direction of the reinforcement(s) and its coupling effects, compression softening of cracked concrete, aggregate interlock, and yielding of the reinforcement. The simulation of post-cracking aspects of reinforced concrete response has received special attention.

- Implement the proposed material model into a special purpose finite element program consisting of solid and degenerate shell elements.

- Implement layering procedures to consider the variation of material properties and geometry, and treat the development and propagation of cracks and other nonlinearities across reinforced concrete cross-sections.
• Implement procedures to compute the transverse shear stress components through the cross-section in conjunction with the layering procedures for inclusion in the material model.

• Evaluate the performance of the analytical model by analyzing a range of test specimens: panels and shear-walls subjected to in-plane loads, beams and slabs subjected to flexural, torsional and shearing loads by comparing them with the experimental results and ACI design procedures.

• This study is limited to a class of problems where the influence of the out-of-plane normal stress is negligible. Contributions of the out-of-plane shear stresses are included in the material model developed in this study.

• This study is limited to a class of problems where only small strains and displacements need be considered.

• Time dependent effects such as creep and shrinkage and thermal effects have not been included.

• The influence of cyclic loading has not been considered in the material model development.

• Perfect bond is assumed between concrete and steel interface.

• Reinforcement is treated as smeared axial layers neglecting its shear contribution.

1.2 Organization

In chapter 2 of this report a brief review of the different constitutive models for concrete employed in various analytical studies is presented. The constitutive model for uncracked concrete and steel has been selected on this basis. A review of techniques that have been used to model various post-cracking nonlinearities in different analytical investigations is discussed.
In chapter 3 the selected constitutive models for uncracked concrete and steel are presented. The implementation of the material model for concrete is then ratified by comparisons of computed and experimental results.

Chapter 4 of this report deals with the constitutive models for cracked concrete and cracked reinforced concrete. Here the formulations for considering the different post-cracking nonlinearities have been developed.

The implementation of the reinforced concrete model in a special purpose finite element program is discussed in chapter 5. This includes the simulation of concrete, steel, and cracks, the iterative solution algorithm, the numerical integration scheme, the cross-sectional layering techniques, and the procedures employed to test the convergence of the nonlinear algorithm.

The capabilities of the developed models in simulating the different response nonlinearities are explored in chapter 6. A number of experiments on different reinforced concrete structural components vis. panels, walls, slabs are analyzed and the results are compared with the test results.

Conclusions based on this study are then presented in chapter 7. Possible extensions for examination in the future are also identified.
Chapter 2

Literature Review

2.1 Introduction

In order to analytically predict the complete inelastic response of reinforced concrete structures up to failure, the employed model must be capable of properly representing the various nonlinearities present. The time-independent nonlinearities include concrete response under compression up to crushing, post-crushing strain softening behavior, reinforcement yielding and post-yield hardening, cracking of concrete, tensile strain-softening response of plain concrete, stiffness contribution of concrete between cracks referred to as ‘tension-stiffening’, aggregate interlock, dowel action, and degradation of cracked concrete compressive stiffness and strength and stiffness with lateral tensile strains. Further, when cyclic or reversed cyclic loading is considered, the model should additionally incorporate progressive degradation of material properties with the number of cycles of load application and also capture the energy loss in each unloading and reloading cycle (hysteretic behavior). In this chapter a brief overview of some of the capabilities of analytical models developed for simulating different response behavior of RC elements to various loads is reviewed. These include the various constitutive models employed for simulating steel and plain concrete responses, procedures for simulating cracks, and the post-cracking nonlinearities considered in different studies.

2.2 Constitutive Modelling of Steel

Constitutive modelling of steel reinforcement has been discussed in detail in ASCE (1982). Typically, a uniaxial stress-strain relationship is employed for steel up to yielding. The post-yield behavior is idealized as either perfectly plastic or hardening plastic. The isotropic hardening parameter is usually defined in terms of
the stiffness at initial yield. In cyclic loading studies, the Bauschinger's effect has been included through the use of kinematic hardening.

To represent the reinforcement in finite element formulations three different approaches have been considered - (1) Smeared layer, (2) embedded bar, (3) discrete bar elements (figure 2.1).

In the smeared layer approach the steel is distributed across the entire concrete element and is oriented in the direction of the steel bar as shown in figure 2.1 (a). In the embedded formulation the steel bar is located inside the concrete element and displacement compatibility between the two materials is enforced at the ends (figure 2.1 (b)). In the discrete element approach steel bar elements are connected explicitly with the surrounding concrete as shown in figure 2.1 (c).

2.3 Constitutive Modelling of Uncracked Concrete

In order to simulate the precracking response of concrete, a number of constitutive models have been proposed in the literature. These can be broadly categorized into the following groups: elasticity based models, plasticity based models, damage mechanics based models and models based on endochronic theory. The ASCE (1982) task committee report, presented an exhaustive survey on the state of the art in reinforced concrete analysis up to that time. This report examined the constitutive models employed for uncracked concrete, the different failure criteria for concrete, the procedures for simulation of cracks, post-cracking nonlinearities considered in different studies, constitutive models for steel, and the type of problems examined using these models. Hence, the emphasis in this survey of literature is on more recent work while a limited number of earlier studies relevant to the present discussion are also included.

2.3.1 Failure Criteria for Concrete

Chen (1982) examined a number of failure criteria employed for concrete and presented the details of the different failure criteria. These include the one parameter envelope (Von Mises criterion), two parameter envelope (Mohr-Coloumb crite-
Figure 2.1: Representation of Steel in Finite Element Analysis of Reinforced Concrete (a) Smeared Steel Layer (b) Embedded Steel Element (c) Discrete Bar Element.
rion), William-Warnke’s three and five parameter envelopes (1975), Ottosen’s four parameter envelope (1977), and Hsieh-Ting-Chen four parameter envelope (1979). These models have been examined through correlations with available test data. Chen’s study concludes that of these different criteria, the four parameter model proposed by Ottosen (1977), the four parameter model proposed by Hsieh-Ting-Chen (1979) and the three and five parameter models proposed by William and Warnke (1975) are the most suitable, capable of properly representing ultimate strengths under general stress states. These models have all the required characteristics of a failure surface vis. smoothness, convexity, symmetry and curved meridians. They also include the one and two parameter envelopes as special cases.

2.3.2 Stress-Strain Response in Concrete

2.3.2.1 Elasticity Based Models

A number of researchers have employed elasticity based models primarily because these are simpler to implement. In these models the nonlinear behavior of concrete is represented by appropriately changing the secant modulus (total formulation) or tangent modulus (incremental formulation). The models by Liu et al. (1972), Darwin and Pecknold (1974), Kotsovos and Newman (1978), Ottosen (1979), Bazant and Tsubaki (1979), Gerstle (1981), Stankowski and Gerstle (1985), etc. are the most notable.

Incremental Formulations

In an incremental formulation the incremental stresses are related to the incremental strains as $\{\Delta \sigma\} = [D]\{\Delta \varepsilon\}$, where $[D]$ is the tangent stiffness matrix. The material parameters used in this procedure depend on the current incremental stress or strain and the stress or strain history up to that point. In general, incremental elasticity based models have the following characteristics:

- This approach is an incrementally linear procedure having path dependent characteristics present in it.
They are simple to implement and can generate the stress-strain behavior of concrete accurately if a broad base of data is available.

Their applicability is restricted to states of stresses for which the material parameters have been calibrated.

Loading, unloading and neutral loading conditions are not properly defined.

They do not treat the post-peak strain softening behavior properly.

A brief review of some incremental elasticity based models follows. Liu et al. (1972) proposed an incremental orthotropic model that accounts for stress induced anisotropy. The material parameters are modified based on total strains. The principal stresses are expressed as closed form expressions in terms of the total strains and the ratio of the principal stresses. The axis of orthotropy is assumed to coincide with the current principal stress direction. Milford and Schnobrich (1984) have employed this model in their study of reinforced concrete cooling towers.

Darwin and Pecknold (1974) proposed an incremental orthotropic model which is based on the concept of equivalent uniaxial strains. Here the effects of biaxial stresses on internal damage in concrete are represented by equivalent uniaxial stress-strain curves in each principal stress direction. The equivalent uniaxial strains are not real strains and do not transform under axis rotations as do true strains and are significant only as a measure on which to base the variations in material properties. The principal stress axis may rotate and need not coincide with the strain direction. The equivalent uniaxial strains are accumulated in the stress directions. This is done without taking into account the axis rotation. They have used this model for the analysis of beams subjected to static and cyclic loadings. Noguchi (1985) has employed this model for concrete in his study of beam-column joints.

The orthotropic models of Liu et al. (1972) and Darwin and Pecknold (1974) have been strongly criticized by Bazant (1979) on both physical and theoretical grounds.

Stankowski and Gerstle (1985) have proposed a simple hypoelastic formulation for concrete based on data from tests under complex load histories. This incremental
formulation relates the octahedral stresses to octahedral strains through the bulk and shear modulii. There is also an interaction between hydrostatic and deviatoric stress components through coupling modulii obtained from experiments. This model has been validated for special axisymmetric loading cases. This formulation is a modification of a model proposed by Gerstle (1981) where the coupling between the hydrostatic and deviatoric stress components was not considered.

**Total Formulation**

In a total formulation the total stresses and total strains are related through the secant stiffness matrix \([D]\) as \(\{\sigma\} = [D]\{\varepsilon\}\). The secant material modulii of concrete are obtained directly from experimental data. In general, total elasticity based models have the following characteristics:

- The use of this approach implies path independent behavior, while concrete exhibits path dependency.
- They are simple to implement and can generate the stress-strain behavior of concrete accurately if a broad base of data is available.
- Their applicability is restricted to states of stresses for which the material parameters have been calibrated.
- Loading, unloading and neutral loading conditions are not properly defined.
- Strain softening response can be simulated with relative ease.

Kotsovos and Newman (1978) proposed a constitutive model for concrete characterized by a total formulation. The two secant material properties, bulk modulus and shear modulus, are evaluated based on curve fitting the data related to concrete compressive strength. The octahedral stresses are related to the octahedral strains through these modulii. A coupling stress which is a function of both the octahedral stress and the deviatoric stress is also defined and employed in computing the octahedral strains.
Ottosen (1979) has developed a secant nonlinear elastic model for concrete. The Young's modulus and Poisson's ratio are modified based on a nonlinearity index. This index relates the current most compressive stress to its ultimate value. The model has been compared with biaxial and triaxial test data for both proportional and non-proportional loading situations and has performed reasonably well. Ottosen has used this formulation in the study of panels, beams and pull-out test specimens.

Bazant and Tsubaki (1979) developed a total strain formulation which includes path dependent parameters. The expressions used are algebraic and independent of loading surfaces or intrinsic time functions. The model has been compared with test data and performs as well as endochronic theories and plastic fracturing models. However, the formulation is not explicit and needs to be modified into its incremental form for implementation and thus is complex.

2.3.2.2 Plasticity Based Models

Plasticity based models have become popular because they are relatively simple to implement, do not depend on a large number of parameters for calibration, and have a sound mathematical basis in their formulation. In these models it is possible to compute the strains due to elastic behavior (fully recoverable) and those due to plasticity (irrecoverable) separately. The recoverable strains are treated within the framework of elasticity theory while the irrecoverable plastic strains are computed based on the theory of plasticity. These models can be further classified on the basis of their hardening rules, as perfectly plastic and hardening plastic models. Hardening plasticity requires two surfaces - the loading surface and the failure surface. Typically the behavior of concrete is assumed to be elastic up to 30% of its uniaxial compressive strength. A loading surface is assumed at this level which is usually assumed to be similar in shape to the failure surface. This loading surface expands with increasing loads based on hardening laws (isotropic, kinematic, mixed) until it meets the failure surface. The components of the plastic strain increments are computed based on assumed flow rules (associated, non-associated).
Chen (1982) has discussed the plasticity based formulations used in RC analysis prior to 1982 in detail. Some of the general characteristics of plasticity based models are:

- They guarantee a stable and unique description of the material law when the normality rule and convexity requirements (Drucker's postulates) are satisfied.
- Loading, unloading and neutral loading conditions are properly defined.
- They account for stress history dependent behavior.
- Residual strains can be computed.
- They do not account for any stiffness degradation.
- Numerical difficulties are encountered when simulating post-peak strain-softening behavior (Glemberg and Samualsson 1983 is an exception).
- They predict an unusually high volume expansion for large hydrostatic compressive stresses and thus represent dilatation poorly.

A brief survey of some of the more recent plasticity based models is presented herein.

Glemberg and Samualsson (1983) have proposed a concrete constitutive model based on the theory of plasticity. This model employs an isotropic strain hardening law up to the ultimate strength. Beyond ultimate strength, the strain softening behavior is also included using exponential functions which define the shrinking of the failure envelope. In their formulation negative definite stiffness tensors are permitted.

Han and Chen (1985) have proposed a non-associated plasticity model. The hardening characteristics of this model account for its ductility in compression and brittleness in tension. The shape function used to define the loading surface is determined such that for triaxial tension no hardening is present, thus representing...
brittle failure. This shape function is designed such that the hardening zone increases with hydrostatic compression. To adequately describe volume contraction and dilatancy, a non-associated plastic flow rule has been employed.

Hu and Schnobrich (1987) have employed a plasticity based model in their study. The formulation includes an isotropic hardening law and a multiple yield criterion based on the stress region. Both associated and non-associated flow rules have been employed along with an equivalent uniaxial stress-strain law.

Torrent et al. (1987) have employed a work hardening plasticity model and failure of concrete under multiaxial stresses. The initial yield surface (loading surface) is closed, while the failure surface remains open along the negative hydrostatic axis, similar to the Ottosen failure criterion (1977). The model shows good agreement with test data.

Channakeshava (1987) has used an elasto-plastic model. This model incorporates isotropic hardening, a segmented equivalent uniaxial stress-strain law, and a modified Mises envelope for its loading and failure surface. In this study it is assumed that plastic stress redistribution follows fracture stress redistribution. An associated flow rule is used to evaluate the plastic strains.

Frantzeskakis (1987) has proposed a model based on a non-associated plasticity flow rule. The yield surface is defined as a Drucker-Prager function. The plastic potential employed to calculate the plastic strains is also a Drucker-Prager function.

Bounding Surface Plasticity Models
Bounding surface plasticity models first evolved to consider the degradation of material properties due to cyclic loading in metals. This approach has since been used to describe the behavior of concrete. A brief review of some of the recent models using this approach is presented here.

Fardis et al. (1983) have used a bounding surface model in stress space proposed by Dafalias and Popov for metals. The bounding surface encloses the current stress point for a given state of stress and strain. This surface is a function of the stress state and the maximum principal compressive strain experienced by the material. The plastic strains are functions of the distance between the current stress point and the bounding surface measured along the instantaneous loading direction.
Buyukozturk and Chen (1985) have proposed a rate independent constitutive model for multiaxial cyclic loading. The concrete is assumed to experience a continuous damage process under load histories. A damage dependent bounding surface in stress space is used to predict the strength and deformational characteristics of the gross material under general loading paths. The size of the bounding surface reduces with increase in damage and the functional dependence of the material modulii on stress/damage gives a realistic picture. Cyclic degradation of properties, post-failure strain-softening and shear compaction dilatancy are also included in this model.

Eberhardsteiner et al. (1987) have proposed a bounding surface plasticity model. In the pre-failure region, the bounding surface is fixed in the stress space. The direction of the plastic flow is determined on the basis of experimental test data. The loading criterion employed by Stankowski and Gerstle (1985) is used in this study.

### 2.3.2.3 Plastic Fracturing Theory

While the treatment of concrete deformation response in pre-failure stress states has been well established through plasticity based models, the constitutive aspects of concrete in the post-failure region has met with some difficulty. For a given uniaxial stress two strain values can be identified from the stress-strain relationship for concrete, one prior to failure and one in the post-failure softening region. Thus, the strains are not uniquely defined for a given general stress state. Moreover, strain-softening response in concrete is due to microcracking which, in contrast to plastic phenomena, is accompanied by a decrease of elastic modulii. Models formulated in strain space have been proposed to consider the softening of the stiffness modulus (e.g. Dougill 1976).

Bazant and Kim (1979) proposed a constitutive model for concrete based on plastic fracture. The inelastic behavior of concrete is assumed to consist of both plastic and fracturing deformations. The plastic strains are irreversible and do not cause a change in the stiffness modulus. The fracture deformations, on the other
hand, are reversible and cause reduction of the stiffness modulus as a result of micro-cracking. The plastic strain increments are based on loading surfaces and flow rules formulated in stress space. Similarly the fracture process is described by using a decrement in stresses along with a fracturing surface described in strain space. One important advantage of the plastic fracturing theory is that, in contrast to incremental plasticity, it gives inelastic responses for stress increments tangential to the loading surface (neutral loading), whereas the classical plasticity theory gives a perfectly elastic response for such load increments, which is not true for concrete.

Han and Chen (1985) have examined this model and found it capable of describing the nonlinear behavior of concrete up to ultimate. However, the use of two loading surfaces makes it very complicated to define criteria for loading, unloading, and reloading of the material. Han and Chen (1985) have employed an internal volumetric work criteria to describe the loading/unloading state of the concrete in their studies. Yamaguchi and Nomura (1985) have employed the plastic fracture model in a study of shear walls subjected to cyclic loads. The results obtained in this study compared well with the experimental response.

2.3.2.4 Endochronic Theory Based Models

Endochronic theories for modelling concrete were first employed by Bazant and Bhat (1976). The central concept in this theory is that of intrinsic time. The increments of intrinsic time, which are non-negative, depend on the total strain increment. The size of the intrinsic time increments controls the inelastic strain increments. Thus the endochronic theory represents a special type of viscoplasticity in which the plastic rate coefficient depends not only on the stress and strain but also the strain rate. With their implementation of this formulation Bazant and Bhat (1976) have shown that a wide range of phenomena can be modeled, e.g., strain softening region, volume dilatation, hydrostatic pressure sensitivity and the strain rate effect. But the number of parameters required to describe this formulation is large compared to elasticity or plasticity based models. A number of refinements to the original formulation have been proposed by Bazant (1980) including a loading
criteria. While incremental elasticity and plasticity based models are incrementally linear, the endochronic theory based models are incrementally nonlinear. However, with the use of loading and unloading criteria the performance of endochronic theory based models has been found to be no better than models using classical plasticity based formulations.

2.3.2.5 Damage Mechanics Based Models

Damage mechanics based models have been proposed in the last few years to consider the microcracking damage taking place even before failure. These models are capable of computing the degradation of stiffness with increasing damage that is not considered in classical plasticity. A few models employing this strategy are presented herein.

Ishikawa, Yoshikawa and Tanabe (1985) have proposed a constitutive model for concrete in terms of a damage tensor. The damage tensor is defined in terms of a 3-D damage field where the damage increase in one direction is used to affect the damage rate in the other two directions. From this a 3-D damage strain tensor is defined to allow for a stress-analysis as an initial strain problem. The model is compared to the model of Kupfer et al. (1973). The stress and strain histories in uniaxial cyclic loading are also considered and reasonably simple expressions are obtained.

Ramtani et. al. (1989) have developed an anisotropic damage model. The anisotropic degradation of the material properties is represented by a second order damage tensor. Inelastic strains are included in the stress-strain relations by coupling the damage to the strains. Evolution laws are associated with each damage variable.

Oliver et al. (1990) have proposed an isotropic damage mechanics based model to simulate the behavior of concrete under different stress-strain conditions with special attention to fracture and strain localization under tension. The model employs a damage criterion formulated in strain space or undamaged stress space.
equivalent strain norm is used to evaluate loading, unloading and reloading. Evolution laws for the damage variables and the damage threshold have been proposed. Comparisons between this model and the conventional anisotropic models using simple tension tests and tension-shear test data have also been made. The model has a simple formulation, is easy to implement, has an explicit integration scheme, and is based on a consistent constitutive theory free from empirical factors.

Borderie et. al. (1990) have proposed a damage mechanics model capable of representing crack opening/closing. This model uses damage coefficients which are internal state variables along with variables describing the state of the crack. The constitutive equations employed are derived based on a thermodynamic irreversible process. The performance of this model has been examined in a two dimensional finite element study of tests on beams subjected to cyclic loading.

2.3.2.6 Comparative Studies of Constitutive Models

In order to evaluate the capabilities of the different constitutive models used for concrete it is necessary to make a comparison of these models using some benchmark problems. A number of such studies have been undertaken and a brief evaluation from two such investigations are presented herein.

Eberhardsteiner et. al. (1987) have examined some of the constitutive models employed for concrete. In their study four models were considered: a nonlinear elasticity based model due to Kotsovos and Newman (1978); a Hypoelasticity based model due to Gerstle and Stankowski (1985); an elasto-plastic model due to Han and Chen (1985); and bounding surface plasticity model proposed by Eberhardsteiner et al. (1987).

The performance of these models in non-proportional and non-monotonic loading cases has been evaluated. The constitutive model proposed by Kotsovos and Newman (1978) was found to be inadequate as it lacked parameters dependent on loading histories. The other models were found to be able to predict the non-proportional loading paths with some success, especially the bounding surface plasticity model due to Eberhardsteiner et. al. (1987).
Torrenti and Djebri (1990) have also presented a comparative study of some other constitutive models employed for concrete. For this study, elasticity based models of Ottosen (1979) and Ahmad and Shah (1986); continuum damage model of Mazars (1989); hypoelastic model of Gerstle (1981); orthotropic model of Torrenti (1987); plasticity based models of Frantzeskakis (1987) and Fardis (1983) have been considered. The implementation of each model in their study has been undertaken without special tuning for particular problems. These models were examined for proportional and non-proportional as well as in two and three dimensional loading conditions. The predicted stress-strain responses using all the models in two dimensional proportional loading cases were found to correlate well with the experimental data of Kupfer et al. (1969). In the case of non-proportional loading, the model proposed by Gerstle (1981) and that of Fardis (1983) were found to be inadequate; the most compressive principal stress direction was found to change compared with the experimental data resulting in lower failure loads. In three dimensional loading cases the incremental model proposed by Gerstle (1981) was satisfactory in reproducing the most compressive principal strains in tests reported by Balmer (1949). But the strains in the other two directions were smaller than their test values. The other models employed were found to underestimate the straining even in the most compressive strain directions. However, they conclude that for most macro analysis any of the models considered here would perform adequately.

2.4 Crack Simulation and Post-Cracking Nonlinearities

2.4.1 Crack Simulation

The representation of cracks in the finite element analysis of concrete structures has been accomplished in two different ways: discrete cracks (Ngo and Scordelis 1967) and smeared cracks (Rashid 1968). In discrete crack modelling the nodes along which the crack can propagate are double or four noded, or even eight noded and are uncoupled when the stresses at a particular node violates the cracking criteria (figure 2.2 (a) and (b)).
Figure 2.2: Representation of Cracks in Finite Element Analysis of Concrete (a) and (b) Discrete Cracks (c) Smeared Cracks.
The use of discrete cracking representations has received only limited acceptance due to the difficulty involved in redefining the mesh topology after the crack formation. Further, the uncoupling of the nodal points destroys the banded structure of the global stiffness matrix, greatly increasing the computational effort required. The inclusion of post-cracking nonlinearities through coupling springs at the 'cracked nodes' requires additional parameters which are difficult to obtain from experimental data. Therefore discrete crack based models are not considered any further in this study. In the smeared crack approach the points which 'crack' are assumed to have a reduced stiffness in the direction normal to the crack plane (figure 2.2 (c)). A number of studies have used this approach stemming from its ease of implementation. Additionally, post-cracking nonlinearities are included in a simple manner. A brief review of some of the smeared crack models is presented next.

Some studies have treated the concrete as isotropic prior to cracking and orthotropic once it cracks. The axes of orthotropy are kept normal and tangential to the crack plane (e.g., Darwin and Pecknold 1974). The crack direction is then considered 'fixed' during subsequent loading stages. The stiffness in the two directions are computed independently and the Poisson's effect is assumed lost. Most formulations include a shear stiffness component which is a fraction of its value prior to cracking. Initiation of new cracks is controlled by specified parameters (threshold angle between any two cracks, etc.). In the presence of anisotropic reinforcement, experimental observations indicate that the crack directions change with the increased load application. In such cases, this modelling approach has been found to be unduly stiff in its predictions and overestimates the ultimate strength of test specimens considerably (Barzegar 1988a).

In an attempt to incorporate the anisotropy induced by cracking, and the effect of anisotropic reinforcement, in some studies (Cope et al. 1981, Gupta and Akbar 1984; Milford and Schnobrich 1984; Noguchi 1985; Maestrini and Gupta 1987; Balakrishnan and Murray 1988; Hu and Schnobrich 1988; Kolleger 1988, Ola and Ottosen 1990) the crack direction has been allowed to 'rotate' with subsequent loading. The crack direction is assumed to be perpendicular to the maximum principal tensile strain direction upon further loading. In this approach previously formed
cracks are ignored, thus the formulation violates the tensorial invariance properties (Bazant 1983).

Another approach in the context of smeared crack modelling considers the cracked concrete to be composed of intact (or uncracked) concrete and microcracks, idealized as springs in series (de Borst and Nauta 1985; Barzegar and Schnobrich 1986; Channakeshava 1987; Rots 1985b, 1988). Each crack interface has a stiffness normal to the crack plane as well as a shear stiffness. This concept permits further non-orthogonal cracks to develop without neglecting the effect of previously formed damage planes. Additionally, it simplifies the monitoring of the crack state namely the crack opening, closing, and reopening.

2.4.2 Post-Cracking Nonlinearities

2.4.2.1 Strain-Softening

The formation and propagation of cracks in plain concrete is a gradual process accompanied by a complete loss of strength and stiffness perpendicular to the crack direction. It has been argued (Bazant and Oh 1983) that a strength based criterion for crack propagation is not objective with respect to finite element mesh refinement. Thus if the mesh were refined in front of a given crack different load increments would be required to advance the crack by a unit length. If such a mesh refinement were continued it would imply crack propagation without any load increment at the limit.

To resolve these issues, energy based criteria employed in fracture mechanics have been advanced (Dougill, 1976; Hillerborg et al., 1976; Bazant and Cedolin 1979, 1983; Willam et al., 1984). In this approach the stiffness of concrete normal to the crack plane, i.e. microcrack stiffness, is gradually reduced with increased straining in that direction. The fracture energy consumed to advance the crack by one unit length is assumed to be a material property. The fracture energy dissipation, which controls the softening process, is a function of the crack band width associated with the damaged sampling point. Thus as the deformation grows the elements around the cracked point begin to unload while the cracked point undergoes strain-
softening. Eventually the crack is fully open and the stresses across the crack drop to zero. This process of strain softening in plain concrete has been discussed in great detail by Willam et al. (1984). A number of studies have focused on capturing the stiffness properties of concrete after cracking. Recent studies have shown (Petersson 1980; Dodds et al. 1982; Bazant and Oh 1983; Willam et al. 1984; Rots et al. 1985, 1988) that the shape of the stress-strain curve after cracking has a significant influence on the propagation of cracks and the post cracking deformational response.

Recently De Borst (1990) has proposed a constitutive model based on Cosserat theory. In this approach the governing field equations include rotational degrees of freedom in addition to the conventional degrees of freedom. This Cosserat continuum model warrants convergence of load deflection curves to a unique solution upon mesh refinement and a finite width of the localized zone. This approach does not employ the ‘characteristic length factor’ which is required in conventional models in order to overcome mesh dependencies when softening is present. The performance of this model has been examined using biaxial tests composed of strain-softening elasto-plastic materials.

### 2.4.2.2 Tension-Stiffening

In reinforced concrete structures, the stiffness contribution of concrete between adjacent cracks, known as ‘tension stiffening’, has been simulated in a number of studies (ASCE 1982; Milford and Schnobrich 1984; Chang et al. 1987; Barzegar and Schnobrich 1988b). Figure 2.3 shows the variation of stresses in a cracked RC element. The assumption of ‘perfect bond’, i.e., the steel and concrete fiber adjacent to the steel have the same strain, is inherent to all models considered in this section. This greatly simplifies the implementation of these models but restricts their range of application in, e.g., large diameter bars in monotonic loading and cyclic loading.

In some models (Milford and Schnobrich 1984; Chang et al. 1987; Barzegar and Schnobrich 1988b) tension-stiffening is considered to be smeared on the concrete in a direction normal to the crack. Rots (1988) treats this stiffness contribution as part of the concrete and artificially increases the fracture energy of plain concrete.
Figure 2.3: Stress Distribution in Cracked Reinforced Concrete.
to accommodate the increased stiffness normal to the crack plane. The disadvantage of treating the stiffness normal to the crack as part of the concrete is in the difficulty experienced in accurately estimating the amount of energy dissipated when cracks are skewed with respect to the reinforcement.

In another formulation (Gilbert and Warner 1977) the concrete contribution between cracks is lumped and considered along the reinforcement direction, assuming the concrete to have lost its stiffness normal to the crack plane. Lumping of the tension-stiffening as part of steel resolves the problem of estimating the termination of tension-stiffening. But the state of stress in concrete is not accurately represented as the tensile stresses are underestimated. Kolleger (1988), treats the tension-stiffening as part of the concrete evaluated in the steel direction and transformed to the principal strain direction ('rotating crack model'). Barzegar and Ramaswamy (1990) treat the tension stiffening as part of the concrete and evaluate it in the steel direction. Such a treatment of tension-stiffening provides a more accurate estimation of the stiffness contribution of concrete after cracking.

### 2.4.2.3 Bond-Slip

The increased application of loads causes the damage zone to expand and a considerable amount of slippage between steel and concrete is possible. This is especially severe under cyclic loading and when large diameter bars are used. Some studies (Floegl and Mang 1982; Rots 1985; Noguchi 1985; Balakrishnan and Murray 1988; Channakeshava 1987; Gupta and Maesterini 1987; Keuser and Mehlhorn 1987; Yashikawa and Tanabe 1986a, 1986b) have developed tension stiffening criterion on the basis of allowing slippage. This approach requires that the steel be modeled as a discrete bar. The interface between the steel and concrete is double noded making it a computationally expensive procedure. The linkage between the steel and concrete at each point is described using springs, whose properties are described by explicit bond shear stress-shear slip assumed for this purpose. However, Channakeshava (1987) has considered bond slip for smeared steel layers. Elwi and Hrudy (1989) have described a method of incorporating bond slip in embedded steel formulations.
2.4.2.4 Aggregate Interlock and Dowel Action

Shear stresses developing on a crack surface cause the principal stress directions to shift during subsequent loading. In a number of previous studies, a fixed fraction of the uncracked concrete shear stiffness has been retained for the shear stiffness of cracked concrete (Darwin and Pecknold 1974; ASCE 1982; Milford and Schnobrich 1984; Gupta and Akbar 1984; Gupta and Maestrini 1987; Barzegar and Schnobrich 1986). Some investigations have considered varying the shear stiffness as a function of the crack normal strains (Al Mahaidi 1979; Balakrishnan and Murray 1986 and 1988; Al-Manseer and Phillips 1987; Razaqpur 1988). Kolleger (1988) has used an explicit expression to model the dowel action of the bars. Some analytical studies have considered the influence of the confinement provided by the reinforcement on the sliding action of the crack surfaces (Bazant and Gambarova 1980; Li Baolu et al. 1989; Yashikawa, Wu and Tsubaki 1989) and the crack dilatation due to this sliding. There is considerable experimental evidence that such an interaction can be significant in shear critical problems. This coupling action (shear-dilatancy) should not be neglected when cyclic loading is considered (Pruijssers 1988).

2.4.2.5 Compression Softening of Cracked Concrete

Experimental studies (Vecchio and Collins 1982, 1986) on RC panels subjected to pure shear have indicated that the presence of cracks reduces both the compressive stiffness and strength parallel to the crack planes. They have developed an empirical relationship relating the tensile strains normal to the crack plane to the ultimate strength of the concrete compressive strut between those cracks. Recently other researchers (Kolleger 1988; Dyngland 1989; Shirai and Noguchi 1989) have reached the same conclusion that the strength of the compressive strut is reduced due to the lateral straining in cracked concrete. However, they conclude that the level of strength and stiffness reduction is no more than 20% of the uniaxial compressive strength and that Vecchio and Collins overestimate this reduction.
2.5 Summary

The distribution of reinforcement in a RC structural component is governed by the nature of forces resisted by them. In compression elements like columns, a few reinforcing bars are employed. While in slabs a large number of small diameter bars are placed across the section to resist flexural and torsional loads. Thus the finite element representation of reinforcement is also governed by the structural element being simulated. Embedded or discrete bars are used in simulating column reinforcement. A smeared layer is more representative of the reinforcement placed in slabs. In the present study the focus is on surface structural elements such as slabs and so a smeared layer procedure is adopted for the simulation of steel. The constitutive aspects of steel modelling considered in this study are discussed in chapter 3.

As discussed in section 2.3 the constitutive modelling of concrete for use in nonlinear analysis of RC structures must have the capability to simulate a number of nonlinearities. While incremental formulations (both elasticity and plasticity based) are in usually able to simulate the response of concrete up to the ultimate load adequately, most of these are incapable of simulating the post-peak strain softening due to numerical difficulties. Thus the stress redistribution and mobilization of alternate load paths are not well defined when these formulations are used in nonlinear analysis of RC structures. However, these models, especially the plasticity based models are capable of capturing the unloading and reloading response and the response under non-monotonic loadings adequately.

While models based on plastic fracturing, endochronic theories and damage mechanics are appealing because these models are capable of capturing most of the observed nonlinearities in concrete, they require a number of parameters to describe them completely. The scatter observed in obtaining even uniaxial properties of concrete e.g. tensile and compressive strength of concrete, makes it very difficult to obtain the various other parameters needed by these models to describe the nonlinear response of concrete.
The major emphasis in the present study is to describe the post-cracking aspects of concrete nonlinearities and thus numerical simplicity of the uncracked concrete modelling has been given precedence. However, the need to adequately describe the behavior of concrete up to and beyond the ultimate load has not been compromised. The constraints placed by these criterion have been adequately satisfied by the Ottosen (1979) nonlinear elastic formulation based on a total formulation. The details of the Ottosen (1979) total secant formulation are presented in chapter 3.

Most analytical studies consider the representation of cracks to be 'smeared' over the sample point. This treatment is simple and the computational requirements are limited. This approach has been found to be suitable for load displacement type 'macro' analysis and the more detailed local stress-strain type 'micro' analysis. It has also been found to represent the localization of cracks with a reasonable level of accuracy (Rots 1988). The treatment of crack rotation via the 'rotating crack' model has come under some criticism because it violates tensorial invariance requirements (Bazant 1983). The more recent multiple non-orthogonal cracking formulation accounts for previously formed damage planes and for the states of these damage planes vis. crack opening, closing and reopening. For this reason a smeared crack formulation based on the later approach is used in this study.

The treatment of post-cracking nonlinearities such as tension-stiffening, bond slip, aggregate interlock, compression strut softening and dowel action have been undertaken in different ways in the analytical studies. However, the formulations employed for incorporating these nonlinearities have some inherent difficulties as discussed. In the present study a new post-cracking formulation for tension-stiffening, aggregate interlock, and softening of the compressive strut in cracked concrete have been proposed. The details of the proposed crack formulation and the treatment of post-cracking nonlinearities are presented in chapter 4.
Chapter 3

Constitutive Modelling

3.1 Introduction

In this chapter the constitutive models for concrete and steel employed in this study are presented. The material model for concrete consists of the uncracked concrete stress-strain relationship up to crushing including the post-peak strain softening behavior. Criteria employed to predict cracking or crushing failure are also presented. Treatment of post-cracking phenomena in concrete is addressed separately in chapter 4. The stress-strain relationship for steel up to yield and the post-yield strain-hardening behavior are also described in this chapter.

3.2 Constitutive Modelling of Uncracked Concrete

3.2.1 Failure Criterion for Concrete

In formulating a failure criteria for concrete under combined states of stresses, a proper definition of failure must first be made. Criteria such as yielding, load carrying capacity, initiation of cracking and extent of deformation have been used to define failure. In this study failure means the stresses cannot continue to increase beyond this point (figure 3.1). Generally, concrete failure can be divided into two types: cracking and crushing. These are characterized by brittleness and ductility respectively. Tensile failures are represented by the formation of a number of cracks and the eventual loss of tensile strength and stiffness normal to the crack plane. While in compression a number of microcracks appear with loss of strength in all directions. In this study a four parameter strength envelope proposed by Ottosen 31

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(1977) has been employed. This strength envelope is expressed in equation 3.1 as:

\[ \frac{A}{f_c^2} + \frac{\sqrt{I_2}}{f_c} + \frac{B}{f_c} - 1 = 0 \]  

(3.1)

Where \( I_1 \) and \( I_2 \) are the first stress invariant and the second deviatoric stress invariant respectively. \( f'_c \) is the uniaxial compressive strength of concrete. \( A \) and \( B \) are parameters, to be determined, and \( \lambda \) is expressed as (Ottosen, 1977):

\[ \lambda = K_1 \cos\left\{ \frac{1}{3} \arccos[K_2 \cos 3\theta] \right\} \]  

\[ \lambda = K_1 \cos\left\{ \frac{\pi}{3} - \frac{1}{3} \arccos[-K_2 \cos 3\theta] \right\} \]  

(3.2)  

(3.3)

where \( K_1 \) and \( K_2 \) are parameters. The four parameters \( A, B, K_1, \) and \( K_2 \) needed to describe the failure surface are evaluated using the following experimental data:

- Uniaxial compressive strength \( f'_c \).
- Biaxial compressive strength of concrete, \( \sigma_{bc} = 1.16f'_c \) corresponding to tests conducted by Kupfer (1969, 1973).
- Uniaxial tensile strength $f_t'$

- The method of least squares has been employed to obtain the best fit of the compressive meridian for $\frac{f_t}{f_t'} \geq -5.0$ to the triaxial test results of Balmer (1949) and Richart (1928) (figure 3.2). The compressive meridian is made to pass through $(\xi, \eta) = (-50.0\text{MPa}, 43.61\text{MPa})$.

Thus, the values of the four parameters are obtained for any concrete based on the uniaxial properties of the concrete used, \textit{vis.} $f'_c$ and $f'_t$. Figure 3.2 shows comparisons of the Ottosen strength envelope with the triaxial and biaxial test data.

The characteristics of the Ottosen failure surface represented by equation 3.1 are:

- Only four parameters are needed to describe the envelope,

- Use of invariants eliminates the need for computing principal stresses to determine failure,

- The surface is smooth and convex except at the vertex,

- The meridians are parabolic and open in the direction of the negative hydrostatic axis,

- The trace of the failure surface on the deviatoric plane changes from a triangular shape (hydrostatic tension) to a nearly circular one with increase in hydrostatic compression,

- It contains several earlier failure criteria as special cases. The Drucker-Prager criterion is obtained by setting $A=0$ and keeping $\lambda$ constant, while the Von Mises criterion is obtained setting $A=B=0$, keeping $\lambda$ constant.

In this study $A$, $B$, $K_1$ and $K_2$ are evaluated from the uniaxial compressive and tensile strengths of concrete and are recomputed with variations in these uniaxial properties, \textit{e.g.} due to initial specimen composition, or due to damage during loading etc.
Figure 3.2: Comparison of Ottosen Strength Envelope with (a) Triaxial and (b) Biaxial Test Data, Ottosen (1977).
Modes of failure
In addition to having a suitable failure criterion it is necessary to identify the mode of failure - cracking or crushing. This is especially important in the tension-compression regions, where cracking can precede crushing. In this study the mixed stress region is divided into a cracking zone and a crushing zone on the basis of the ratio of the maximum to minimum principal stresses as:

\[
\text{cracking : } \infty \leq \frac{\sigma_1}{\sigma_3} \leq -0.7333 \frac{f_t}{f_c} \\
\text{crushing : } -0.7333 \frac{f_t}{f_c} \leq \frac{\sigma_1}{\sigma_3} \leq 0.0
\]

The influence of the intermediate stress, \( \sigma_2 \), is neglected. The criterion is similar to the one adopted by Barzegar and Schnobrich (1986) for two-dimensional applications. The principal stresses employed in equations 3.4 and 3.5, \( \sigma_1 \), \( \sigma_2 \) and \( \sigma_3 \) are arranged in descending order (\( \sigma_1 \geq \sigma_2 \geq \sigma_3 \)). The behavior of concrete up to and after crushing is discussed in the next section. The post-cracking response of concrete is discussed in chapter 4.

3.2.2 Stress-Strain Response of Uncracked Concrete

In the pre-cracked phase, the stress-strain relationship for plain concrete is determined using a 3-D constitutive model proposed by Ottosen (1979). In this isotropic nonlinear elastic model the total stresses are related to the total strain as follows:

\[
\begin{pmatrix}
\sigma_x \\
\sigma_y \\
\sigma_z \\
\tau_{xy} \\
\tau_{yz} \\
\tau_{xz}
\end{pmatrix} = \frac{E_{co}}{(1-2\nu)(1+\nu)} \begin{pmatrix}
1-\nu & \nu & \nu & 0 & 0 & 0 \\
\nu & 1-\nu & \nu & 0 & 0 & 0 \\
\nu & \nu & 1-\nu & 0 & 0 & 0 \\
0 & 0 & 0 & 1-2\nu & 0 & 0 \\
0 & 0 & 0 & 0 & 1-2\nu & 0 \\
0 & 0 & 0 & 0 & 0 & 1-2\nu
\end{pmatrix} \begin{pmatrix}
\varepsilon_x \\
\varepsilon_y \\
\varepsilon_z \\
\gamma_{xy} \\
\gamma_{yz} \\
\gamma_{xz}
\end{pmatrix}
\]

where \( E_{co} \) and \( \nu = \nu_{co} \) are the secant stiffness modulus and Poisson's ratio respectively for concrete. These modulii are functions of the states of stress and are
modified as explained in the following sections. This model is capable of simulating the unloading and reloading response of concrete along a secant path as shown in figure 3.3.

![Stress-Strain Diagram](image)

**Figure 3.3: Unloading and Reloading Response of Concrete Along a Secant Path.**

### 3.2.2.1 Computation of Pre-Peak Secant Stiffness Modulus $E_{co}$

The secant stiffness of concrete $E_{co}$ employed in the model is computed (Ottosen 1979) as:

$$E_{co} = \frac{E_i}{2} - \gamma \left(\frac{E_i}{2} - E_f\right) \pm \sqrt{\left[\frac{E_i}{2} - \gamma \left(\frac{E_i}{2} - E_f\right)\right]^2 + E_f^2 \gamma [D(1 - \gamma) - 1]}$$  \hspace{1cm} (3.7)

where the positive sign is taken for the ascending branch (pre-peak) and the negative sign for the descending branch (post-peak). Here $E_i$ is the initial secant stiffness, $E_f$ is the stiffness at triaxial compressive failure, $\gamma$ is the nonlinearity index associated with the proximity of the stress-point to the failure surface and $D$ is a post-peak softening parameter.
The nonlinearity index, $\gamma$, is a measure of the actual stresses with respect to the failure envelope (figure 3.4) and is defined as (Ottosen, 1979):

$$\gamma = \frac{\sigma_3}{\sigma_{3f}}$$

(3.8)

Here $\sigma_3$ is the actual minimum principal stress (compressive) and $\sigma_{3f}$ is the corresponding failure stress value, provided that the other principal stresses, $\sigma_1$ and $\sigma_2$ remain unchanged ($\sigma_1 \geq \sigma_2 \geq \sigma_3$). Thus $\gamma < 1$, $\gamma = 1$ and $\gamma > 1$ correspond to stress states inside, on, and outside the failure surface, respectively.

Ottosen (1979) has proposed a modification to this criterion when tensile stresses are present, since the concrete behavior is more linear with increased tensile stresses. When at least one principal stress, $\sigma_1$, is tensile the stress state is modified by superposing a hydrostatic pressure, $-\sigma_1$ on it, thereby producing a biaxial compressive stress state $(\sigma'_1, \sigma'_2, \sigma'_3) = (0, \sigma_2 - \sigma_1, \sigma_3 - \sigma_1)$. The nonlinearity index

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\( \gamma \) is:

\[
\gamma = \frac{\sigma_3'}{\sigma_{3f}'} \tag{3.9}
\]

where \( \sigma_{3f}' \) is the failure stress state provided that \( \sigma_1' \) and \( \sigma_2' \) are unchanged. This approach reduces the \( \gamma \) value reflecting linearity (equation 3.9) and \( \gamma \leq 1 \) always holds.

The failure stiffness \( E_f \) is a function of the loading, type of concrete etc. and is computed as (Ottosen, 1979):

\[
E_f = \frac{E_c}{1 + 4(A - 1)X} \tag{3.10}
\]

in which \( E_c = f_c'/\epsilon_c' \) is the stiffness at peak uniaxial compressive stress, and \( A = E_i/E_c \). In equation 3.10, \( X \) represents the dependence on the actual loading and is expressed as:

\[
X = \left\{ \frac{\sqrt{J_2}}{f_c'} \right\} + \frac{1}{\sqrt{3}} \tag{3.11}
\]

where \( J_2 \) is calculated from the same failure stress state which is used to evaluate \( \gamma \), and is expressed as:

\[
J_2 = \frac{1}{3} \left\{ \sigma_1^2 + \sigma_2^2 + \sigma_{3f}'^2 - \sigma_1 \sigma_2 - \sigma_2 \sigma_{3f}' - \sigma_1 \sigma_{3f}' \right\} \tag{3.12}
\]

When tensile stresses are present, \( E_f = E_c \) is assumed.

The parameter \( D \) (equation 3.7) determines the post peak stress-strain curves. However, there are limitations in selecting \( D \) if the stress-strain curves should reflect the following requirements (Ottosen 1979):

- An increasing function with no inflexion points prior to failure.
- A decreasing function with at most one inflexion point after failure.
- A residual strength of zero after very large strains.

To achieve these features \( A \geq 4/3 \) is retained (equation 3.10). A further restriction on \( D \), enforced by the following relations, is:

\[
(1 - \frac{A}{2})^2 \leq D \leq 1 + A(A - 2) \quad A \leq 2 \tag{3.13}
\]

\[
0 \leq D \leq 1.0 \quad A \geq 2 \tag{3.14}
\]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
3.2.2.2 Post-Peak Stiffness Computation

The post-peak behavior where some small stresses remain is modeled by the following procedure proposed by Ottosen (1979). At peak (point A in figure 3.5) the stresses result in a nonlinearity index $\gamma_f \leq 1$. In figure 3.5 the curve AB is obtained by shifting the descending curve MN parallel to the horizontal axis. The secant stiffness in the post-peak region is expressed as:

$$E_{co} = \frac{\gamma E_{MN} E_A E_M}{\gamma E_A E_M + \gamma_f E_{MN}(E_M - E_A)}$$  \hspace{1cm} (3.15)$$

$E_{MN}$ is obtained using the actual $\gamma$ (current stress state) along with the negative sign in equation 3.7. $E_A$ and $E_M$ are the ascending (pre-peak) and descending (post-peak) stiffnesses computed using $\gamma = \gamma_f$, the value of $\gamma$ at ultimate, along with a positive and negative sign, respectively, in equation 3.7. Thus, there is a gradual change in the post-peak behavior with changing stress states.

Figure 3.5: Computation of Secant Stiffness, $E_{co}$ in the Post-Peak Region.
3.2.2.3 Changes in Poisson's Ratio $\nu_{co}$

For uniaxial, biaxial, and triaxial compression in concrete the volumetric change is initially that of compaction followed by dilatation. Thus, the Poisson's ratio has been generalized for multiaxial states of stress by Ottosen (1979) in terms of the nonlinearity index $\gamma$ as (figure 3.6):

$$\nu_{co} = \nu_i \quad \gamma \leq \gamma_a$$

$$\nu_{co} = \nu_f - (\nu_f - \nu_i) \sqrt{1 - \left[\frac{\gamma - \gamma_a}{1 - \gamma_a}\right]^2} \quad \gamma \geq \gamma_a$$

(3.16)

(3.17)

$\nu_i$ and $\nu_f$ are the initial secant Poisson’s ratio and its value at peak compressive stress respectively. The parameter, $\gamma_a$ has a value of 0.80. The second expression is valid only up to failure but it is assumed that dilatation continues beyond failure. In the post-peak zone, $\nu_{co}$ is modified along with $E_{co}$. However, this change is related to the bulk modulus at failure, $K_f$, which is kept constant beyond the peak stress. For a given post-peak stiffness $E_a$, a Poisson’s ratio $\nu^*$ is obtained with the bulk modulus, $K_f$, remaining unchanged. The secant Poisson’s ratio is then obtained as (Ottosen, 1979):

$$\nu_{co} = 1.005\nu^*$$

(3.18)

The Poisson’s ratio $\nu_{co} < 0.5$ is always maintained. The value of the parameter $\nu_f$ (Ottosen, 1979) is:

$$\nu_f = 0.36$$

(3.19)

3.2.2.4 Input Parameters

The input parameters required for this material model are the initial secant stiffness $E_i$, the uniaxial compressive strength $f'_c$, $\epsilon_c$, the strain at peak uniaxial compressive stress, the initial Poisson’s ratio $\nu_i$, and the post-peak softening parameter D. These quantities are all user supplied values based on appropriate test data.
3.2.3 Correlations with Test Results

The failure envelope and stress-strain relationships discussed in the preceding sections have been implemented in a special purpose finite element program. The details of the implementation are discussed in chapter 5. The implementation of the concrete model has been ratified by comparing the computed results obtained in the present study with the test results (Linse and Aschl 1976, Schickert and Winkler 1977) and also analytical results obtained by Ottosen (1981, 1982). The analytical results obtained by Ottosen (1981, 1982) are the only ones included for comparisons because they provide a means of ratification for its implementation in the present study.

In figure 3.7 stress-strain response from a uniaxial compression test along with the results obtained in the present study are shown; results obtained by Ottosen (1981, 1982) have also been included. The uniaxial material properties employed in the various analyses are presented on the figures. Figures 3.8 and 3.9 shows
the comparisons of triaxial stress-strain response along the compression meridian
(loading path as indicated on the figures) with the test results; results obtained by
Ottosen (1981,1982) are similar to the results obtained in the present study. Triaxial
stress-strain responses along the tensile meridian (Loading path as indicated on the
figure) are shown in figure 3.10.

![Figure 3.7: Comparison with Uniaxial Compressive Loading Test Data.](image)

While all of the above examples represent monotonic proportional loadings,
Ottosen (1981,1982) has shown that the material model performs adequately for
certain classes of non-proportional loadings as well. Figure 3.11 shows results from
a test where the hydrostatic loads are constant throughout the loading. However,
after a certain hydrostatic stress level (-25.5 MPa), the load is applied along the
compressive meridian while keeping the hydrostatic stress constant. The analyt-
tical results obtained in this study are similar to the results obtained by Ottosen.
Figure 3.8: Triaxial Loading Along the Compressive Meridian.

Figure 3.9: Triaxial Loading Along the Compressive Meridian.
(1981, 1982) and compare well with the test results.

The analytical predictions obtained in the present study compare well with the experiments and also the predictions obtained by Ottosen (1981, 1982). The dominant nonlinear response characteristics appear to be well represented by Ottosen's model employed in the present study.

### 3.3 Stress-Strain Response of Steel

The steel reinforcement is assumed to be smeared into equivalent membrane layers with uniaxial properties oriented in each reinforcing direction. An elasto-plastic response with possible strain-hardening, identical in tension and compression, is assumed (figure 3.12). The constitutive matrix in an orthogonal coordinate system ($x', y', z'$) with the bar direction coinciding with the $x'$ is:
The modulus of elasticity $E_s$ is modified based on the following expressions:

$$ E_s = E_y \quad |\epsilon_s| < |\epsilon_y| $$  \hspace{1cm} (3.21)

$$ E_s = \alpha E_y + \frac{\sigma_y}{\epsilon_s} |1 - \alpha| \quad |\epsilon_s| \geq |\epsilon_y| $$  \hspace{1cm} (3.22)

Where $\sigma_y$ is the yield stress for steel, $E_y$ the initial secant stiffness, and $\alpha$ is a strain-hardening parameter expressed as the ratio of hardening stiffness to the initial secant stiffness.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 3.12: Stress-Strain Response of Steel
Chapter 4

Post-Cracking Response of Plain and Reinforced Concrete

4.1 Introduction

The weakness of concrete in carrying tensile stresses and the resulting formation of cracks is a major factor in the nonlinear response of reinforced concrete structures. After cracking, strain softening causes the strength and stiffness of plain concrete to gradually reduce in the direction normal to the cracking plane. The load carrying capacity and the stress distribution are significantly altered due to this effect.

After cracking has occurred in reinforced concrete, the tension-stiffening in the vicinity of the reinforcement enhances the stiffness and stress carrying capacity of concrete near the tension region. The shear stress transferred along the crack interfaces is controlled by the aggregate interlock mechanism. The crack opening during the sliding action, referred to as shear dilatancy, is resisted by the reinforcement crossing the crack. Further, the sliding action along the cracked surface is directly resisted by the reinforcement through the dowel action. Due to cracking and the release of confining effect in the crack normal direction, concrete compressive strength and stiffness parallel to the crack direction diminishes.

In this chapter the post-cracking behavior of plain and reinforced concrete is described and material models to simulate these phenomena are proposed. To facilitate this development, rheological models representing the various phenomena are described.
4.2 Plain Concrete Under Uniaxial Tension

A plain concrete element subjected to a general state of stress, at incipient cracking is shown in figure 4.1. In formulating the post cracking concrete behavior, an approach similar to the one used by deBorst and Nauta (1985) is adopted. In this approach the total strain $e_n'$ is resolved into intact concrete strains $e_n^o$ and crack strains $e_n^c$, i.e. $e_n' = e_n^o + e_n^c$. The constitutive matrix for the intact concrete $[D^o]$ is given in equation 3.6 while that for the crack (deBorst and Nauta, 1985) is:

![Figure 4.1: Plain Concrete Element at Incipient Cracking](image)
where \( E_{nn}^{cr} \) is the crack stiffness normal to the crack plane (n) and \( G_{ns}^{cr} \) and \( G_{nt}^{cr} \) are the crack interface shear stiffness components along the crack plane (s-t). The crack normal stiffness is determined from the bilinear stress-strain curve shown in figure 4.2(e), where \( G_f \) is the fracture energy, assumed to be a material constant. The formulation for the crack interface shear stiffness is discussed in section 4.3.4.

The post-cracking constitutive matrix of plain concrete in the global \((x,y,z)\) coordinate system due to strain softening is formulated as (de Borst and Nauta, 1985):

\[
[D]_{x,y,z}^{cr-con} = [D]^{co} - [D]^{co}[N][([D]^{cr} + [N]^T[D]^{co}[N])^{-1}[N]^T[D]^{co}}
\]  

(4.2)

where \([N]\) represents the transformation of the crack local strains to the global directions given by:

\[
\begin{pmatrix}
 l_x^2 & l_x l_y & l_x l_z \\
 m_x^2 & m_x m_y & m_x m_z \\
 n_x^2 & n_x n_y & n_x n_z \\
 2l_x m_x & l_x m_y + l_y m_x & l_x m_z + l_z m_x \\
 2m_x n_x & m_x n_y + m_y n_x & m_x n_z + m_z n_x \\
 2n_x l_x & n_x l_y + n_y l_x & n_x l_z + n_z l_x
\end{pmatrix}
\]  

(4.3)

where \( l_x, m_x \) and \( n_x \) are the direction cosines of the normal to the crack plane (n) with respect to the global directions \((x,y,z)\). Similarly \( l_y, m_y \) and \( n_y \) and \( l_z, m_z \) and \( n_z \) represent the direction cosines of two orthogonal directions (s-t), on the crack plane.

For the plain concrete tensile specimen shown in figure 4.2 (a), using equation 4.2, the post-cracking stiffness matrix can be shown to be (assuming the crack...
Figure 4.2: (a) Cracked Plain Concrete Specimen, (b) Rheological Representation of Plain Concrete Cracking, (c) Stress-Strain Relationship For Cracked Plain Concrete, (d) Stress-Strain Relationship for Uncracked Concrete, (e) Stress-Strain Relationship for Cracks.
normal is along the global x direction):

\[
[D]_{x,y,z}^{con} = E'' \begin{pmatrix}
(1-\nu)\mu A & \nu B & 0 & 0 & 0 \\
0 & D & 0 & 0 & 0 \\
0 & 0 & D & 0 & 0 \\
0 & 0 & 0 & (1-2\nu) \beta & 0 \\
0 & 0 & 0 & 0 & (1-2\nu) \beta
\end{pmatrix}
\]

(4.4)

where:

\[
E'' = \frac{E_{co}}{(1-2\nu)(1+\nu)}
\]

(4.5)

\[
A = 1 - \nu - 2\nu^2
\]

\[
B = 1 - \nu - 2\mu^2
\]

\[
D = (1-2\nu)(1-\mu^2)
\]

\[
F = (1-2\nu)(1+\mu^2)
\]

The parameter \( \mu \) represents the degree of damage in the secant stiffness of concrete and \( \beta \) represents the amount of damage in the shear stiffness. When the crack is fully open (\( \mu=0.0 \)), the stiffness normal to the crack and the coupling stiffness relating the cracked and uncracked directions disappear. A rheological representation of the plain concrete tensile specimen is shown in figure 4.2 (b). Figure 4.2 (c), (d), (e) show the stress-strain relationships of the composite cracked concrete, the intact concrete and the crack, respectively.

### 4.3 Modelling of Cracked Reinforced Concrete

#### 4.3.1 RC Bar Subjected to Uniaxial Tension

In establishing the stiffness of a simple RC bar element figure 4.3(a), tension-stiffening represents the ability of the intact concrete between adjacent cracks to transmit tensile stresses. In this case, the post-cracking response is dominated by the presence of the reinforcement. One possible rheological model is depicted in figure 4.3 (b). Here, the fracture energy \( G_f \) of plain concrete has been artificially
increased by $\Delta G_f$. This increase in energy dissipation capacity represents an average strain energy associated with the concrete-reinforcement interface bond stresses which cause distributed cracking. The value of $\Delta G_f$ is usually much higher than $G_f$, which implies that the tension stiffening effect dominates the energy dissipation characteristics of the cracked reinforced concrete element.

![Diagram](https://example.com/diagram.png)

Figure 4.3: (a) RC Element, (b) Rheological Representation of Reinforced Concrete Cracking, Rots (1988)

For the RC tensile specimen the curve representing the combined effects of tension softening and tension stiffening may be constructed based on measured post cracking responses of RC test specimen under uniaxial tension. But, in cases where the cracks are non-orthogonal to the reinforcing direction, or where crack
directions shift and additional crack formation is possible, this procedure meets with difficulty. This is because the strain softening behavior is dominant in the crack normal direction, while the tension-stiffening is effective in the reinforcement direction(s).

To rectify this shortcoming, an alternate rheological model is proposed as shown in figure 4.4. It is assumed that on cracking, the normal stiffness in the microcrack-intact concrete assembly is completely lost - $E_{co}/1000$ is used for numerical stability purposes. The tension-stiffening spring stiffness $E_t^{ts}$ which now models the combined stiffness of both the bond stresses and the strain-softening effects is activated when the strain in the reinforcing direction exceeds the uniaxial cracking strain $\epsilon_{cr}$. The constitutive matrix for the cracked concrete specimen $[D]^{cr-rc}_{x,y,z}$ including tension stiffening is then:

$$[D]^{cr-rc}_{x,y,z} = [D]^{cr-con}_{x,y,z} + [T]^T[D]^{ts}[T]$$

where:

$$[D]^{ts} = \begin{pmatrix}
E_t^{ts} & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0
\end{pmatrix}$$

and $[T]$ transforms $[D]^{ts}$ from the steel direction $(x', y', z')$ to the global direction $(x, y, z)$ given by:

$$[T] = \begin{pmatrix}
I_x^2 & m_x^2 & n_x^2 & l_x m_x & m_x n_x & n_x l_x \\
I_y^2 & m_y^2 & n_y^2 & l_y m_y & m_y n_y & n_y l_y \\
I_z^2 & m_z^2 & n_z^2 & l_z m_z & m_z n_z & n_z l_z \\
2l_x l_y & 2m_x m_y & 2n_x n_y & l_x m_y + l_y m_x & m_x n_y + m_y n_x & n_x l_y + n_y l_x \\
2l_x l_y & 2m_x m_y & 2n_x n_y & l_x m_y + l_y m_x & m_x n_y + m_y n_x & n_x l_y + n_y l_x \\
2l_z l_y & 2m_z m_y & 2n_z n_y & l_z m_y + l_y m_z & m_z n_y + m_y n_z & n_z l_y + n_y l_z \\
2l_z l_y & 2m_z m_y & 2n_z n_y & l_z m_y + l_y m_z & m_z n_y + m_y n_z & n_z l_y + n_y l_z
\end{pmatrix}$$

$l,m,n$ are direction cosines relating the tension stiffening directions (same as the steel directions) to the global directions.

With this procedure the memory of cracking is retained and possible changes in the shear interface stiffness may be taken into account. Additionally, due to establishment of the tension stiffening spring in the direction of the reinforcement, the

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 4.4: Rheological Representation of Reinforced Concrete Cracking, Present Study

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
stiffness contribution of the bond stresses are modeled in their proper orientation. The shape of the tension-stiffening spring (figure 4.4) is determined by analyzing the results of number of a number of tests on RC specimen. The curve is expressed in terms of known quantities vis. cracking strength of concrete and yield strain of the reinforcement. The values of $\alpha, \beta, \gamma, \delta$ are established through comparison of the tensile stress-strain response with uniaxial tensile test results (presented in section 6.2.1).

It should be noted that when the crack normal stiffness component is completely lost in the manner described above, the cracked reinforced concrete assembly does not retain any Poisson’s effect. This is easily seen form equation 4.4. However, experimental evidence presented by Kollegger (1988) and Dyngland (1989) indicate that after cracking some cross coupling stiffness is still present; they observed that for a previously cracked RC element, compressive loading in the uncracked orthogonal direction caused significant straining in the cracked direction. This coupling disappears gradually with increasing damage in either direction due to loading. In the present study this coupling effect is included and is expressed in terms of the tension-stiffening stiffness components in the direction of the reinforcement and is expressed as:

$$
[D]^{Coupling}_{\text{CrossCoupling}} = 
\begin{pmatrix}
0 & \nu\sqrt{E_1 E_2} & \nu\sqrt{E_1 E_3} & 0 & 0 & 0 \\
\nu\sqrt{E_1 E_2} & 0 & \nu\sqrt{E_2 E_3} & 0 & 0 & 0 \\
\nu\sqrt{E_1 E_3} & \nu\sqrt{E_2 E_3} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

(4.8)

where $E_1, E_2$ and $E_3$ are normal stiffness components in three mutually orthogonal directions including atleast one reinforcement direction. This coupling matrix is transformed to the global ($x,y,z$) directions and then added to the cracked concrete stiffness matrix expressed by equation 4.6 to result in the stiffness matrix of the RC bar as:

$$
[D]^{cr-con}_{\text{CrossCoupling}} = [D]^{cr-con}_{\text{CrossCoupling}} + [T]^T[D]^{cr-con}_{\text{CrossCoupling}}[T] + [T]^T[D]^{Coupling}_{\text{CrossCoupling}}[T]
$$

(4.9)
For example, for the uniaxially reinforced bar shown in figure 4.4 (a) these terms are $E_1 = E'_{1s}$, where $E'_{1s}$ is the secant stiffness of the tension stiffening spring along the steel direction (equation 4.7) and $E_2 = E_3 = E^{co}$, Where $E^{co}$ is the stiffness of the concrete in the remaining two unreinforced directions provided that the strains in these two directions are compressive. Tensile strains in an unreinforced direction would imply an absence of coupling as in the plain concrete response. According to equation 4.8 the coupling stiffness terms are present until either tension-stiffening effect is exhausted completely or the concrete crushes.

4.3.2 General Triaxial Loading of RC

In the preceding section the behavior of a uniaxially reinforced concrete bar with the crack orthogonal to the bar direction was discussed. However, in most practical situations the orientation of the cracks are at some skew angle with the reinforcement. In such situations, it is necessary to compute the tension-stiffening spring parameters while accounting for the this skew angle. Further, it is possible for a cracked sampling point to contain steel in three orthogonal orientations and it is then necessary to compute the tension-stiffening contribution in each steel direction accordingly. Additionally, the coupling stiffness expressed by equation 4.8 must also account for these tension-stiffening contributions in their proper directions. In this section RC elements with three different cases of orthogonal arrangements of reinforcement are considered.

4.3.2.1 Tri-directional Orthogonally Reinforced Concrete Elements

Consider a hypothetical RC element subjected to a general state of loading (as shown in figure 4.5). For the purpose of this discussion the element has been considered to be anisotropically reinforced in three orthogonal directions $(x', y', z')$. For the given load, the equilibrium equations in the three reinforcement directions $(x', y', z')$ at incipient cracking are:

$$
\sigma_{x'} Al' + \rho_{x'} \sigma_{x'} Al' + \tau_{x'y'} Am' + \tau_{x'z'} An' = \sigma_{nn} Al' \\
(4.10)
$$
Figure 4.5: Tri-Directionally Reinforced Concrete Element Subjected to General Loads Prior to Cracking
where \((l'_1, m'_1, n'_1), (l'_2, m'_2, n'_2)\) and \((l'_3, m'_3, n'_3)\) are the direction cosines of the three principal stress directions with respect to the three reinforcement directions. \(A\) represents the area of the element along the crack plane. The rheological representation of this element at this stage and its schematic representation are shown in figure 4.6 (a) and (b), respectively.

Figure 4.6: (a) Rheological Representation of a Tri-Directionally Reinforced Concrete Element Prior to Cracking (b) Schematic Representation of the Element.

When a small increment of load is applied to the specimen and it cracks, it is initially assumed that the shear stresses developing on the crack plane are negligible (the presence of reinforcement in each direction ensures a gradual build-up of the shear stress). Figure 4.7 shows the state of stress in the RC element under these conditions. The equilibrium equations in the three reinforcing directions are:

\[
\sigma_y' A m'_1 + \rho_y' \sigma_{xy}' A m'_1 + \tau_x'y A l'_1 + \tau_y' A n'_1 = \sigma_{nn} A m'_1
\]  
\[
\sigma_x' A n'_1 + \rho_x' \sigma_{xx}' A m'_1 + \tau_x' A l'_1 + \tau_x'y A n'_1 = \sigma_{nn} A n'_1
\]
Figure 4.7: Tri-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Strain $\epsilon < \epsilon_{cr}$ in the Steel Direction.
A gradual strain softening behavior in the crack normal direction is simulated by a strain-softening spring in that direction. The assumed shape of the stress-strain response of this spring is shown in figure 4.8. The strains in each steel direction is monitored until at least one of these reaches the uniaxial cracking strain level. At this stage the strain softening spring stiffness in the crack normal direction is released completely and the ‘tension-stiffening’ spring in each steel direction is activated. If the strain in a particular steel direction is below the cracking strains, tension-stiffening is still considered to be activated in that direction if these strains are tensile. If the crack orientation with respect to the reinforcement is such that the strain in a steel direction is compressive no tension stiffening spring is activated in the corresponding steel direction. Figure 4.9 shows the state of stress in the RC element after the tension-stiffening springs are activated. The incremental equilibrium equations at this state of stress is expressed by the following equations:

\[
(\sigma_{x'} + \Delta\sigma_{x'})Al'_1 + \rho_{x'}(\sigma_{sx'} + \Delta\sigma_{sx'})Al'_1 + (\tau_{x'y'} + \Delta\tau_{x'y'})Am'_1 +
(\tau_{x'y'} + \Delta\tau_{x'y'})An'_1 = (\sigma_{nn} + \Delta\sigma_{nn})Al'_1
\] (4.13)

\[
(\sigma_{y'} + \Delta\sigma_{y'})Am'_1 + \rho_{y'}(\sigma_{sy'} + \Delta\sigma_{sy'})Am'_1 + (\tau_{x'y'} + \Delta\tau_{x'y'})Al'_1 +
(\tau_{y'y'} + \Delta\tau_{y'y'})An'_1 = (\sigma_{nn} + \Delta\sigma_{nn})Am'_1
\] (4.14)

\[
(\sigma_{y'} + \Delta\sigma_{y'})An'_1 + \rho_{y'}(\sigma_{sy'} + \Delta\sigma_{sy'})An'_1 + (\tau_{y'y'} + \Delta\tau_{y'y'})Al'_1 +
(\tau_{y'y'} + \Delta\tau_{y'y'})Al'_1 = (\sigma_{nn} + \Delta\sigma_{nn})An'_1
\] (4.15)

\[
\Delta\sigma_{x'}Al'_2 + \rho_{x'}\Delta\sigma_{sx'}Al'_2 + \Delta\tau_{x'y'}Am'_1 + \Delta\tau_{x'y'}Am'_2 = \\
\Delta\sigma_{nn}Al'_2 + \Delta\sigma_{nn}Al'_2 + \Delta\tau_{nn}Al'_3
\] (4.16)

\[
\Delta\sigma_{y'}Am'_3 + \rho_{y'}\Delta\sigma_{sy'}Am'_3 + \Delta\tau_{y'y'}Al'_1 + \Delta\tau_{y'y'}Al'_3 = \\
\Delta\sigma_{nn}Am'_3 + \Delta\sigma_{nn}Am'_3 + \Delta\tau_{nn}Am'_3
\] (4.17)
Figure 4.8: Strain-Softening Envelope Employed Normal to the Cracks in Cracked RC, With Steel Direction Strains $\varepsilon < \varepsilon_{cr}$.
Figure 4.9: Tri-Directionally Reinforced Concrete Element Subjected to General loads, With Atleast One Strain $\varepsilon \geq \varepsilon_r$ in the Steel Direction.
\[ \Delta \sigma_s' A n_1' + \rho_s' \Delta \sigma_t' A n_1' + \Delta \tau_{s't'} A m_1' + \Delta \tau_{s't'} A l_1' = \Delta \sigma_{nn} A n_1' + \Delta \tau_{sn} A n_2' + \Delta \tau_{nt} A n_3' \] (4.18)

It should be clear to the reader that the state of stress in the RC element shown in figure 4.7 and figure 4.9 are equivalent and that the crack normal stress can be identically replaced by an equal stress in each reinforcing spring direction. This is also evident from the above equations. Figure 4.10 (a) and (b) show the rheological representation of the element at this stage of the straining and the corresponding schematic representation of the cracked point, respectively.

Figure 4.10: (a) Rheological Representation of a Tri-Directionally Reinforced Concrete After Full Crack Opening (b) Schematic Representation of a Cracked Point.

The following observations can be made about this approach to modelling tension stiffening:
• The tension stiffening in each reinforcing direction is thus fully defined in
terms of the crack initiating stress and strain and the yield strain of the steel
in the corresponding direction.

• A larger straining in a weakly reinforced direction is directly translated into
rapid release of the tension-stiffening retained in that direction. However,
such a directional loss of tension-stiffening (along with yielding of the steel)
does not translate into a complete loss of tension-stiffening.

• The shear stresses generated on the crack plane immediately after cracking
in such a case is negligible, provided neither of the reinforcements yield at
cracking. If the reinforcement in one direction yields prior to the others,
shear stresses do develop on the crack surface to equilibrate the unbalanced
applied loads in that direction.

4.3.2.2 Bi-Directional Orthogonally Reinforced Concrete Elements

A large number reinforced concrete structural elements have planar geometry.
Consequently, the reinforcement arrangement in these elements is typically a mesh
reinforcement in the plane of the RC element. Consider such an element under a
state of general loading (figure 4.11). The equilibrium equations (along \(x', y', z'\))
for this element at incipient cracking are:

\[
\sigma_{x'} A_{1} + \rho_{x'} \sigma_{x'x'} A_{1} + \tau_{x'y'} A_{m_{1}} + \tau_{z'} A_{n_{1}} = \sigma_{nn} A_{l_{1}}
\]  
(4.19)

\[
\sigma_{y'} A_{m_{1}} + \rho_{y'} \sigma_{y'y'} A_{m_{1}} + \tau_{x'y'} A_{l_{1}} + \tau_{z'} A_{n_{1}} = \sigma_{nn} A_{m_{1}}
\]  
(4.20)

\[
\sigma_{z'} A_{n_{1}} + \tau_{y'z'} A_{m_{1}} + \tau_{z'z'} A_{l_{1}} = \sigma_{nn} A_{n_{1}}
\]  
(4.21)

where \((l_{1}', m_{1}', n_{1}')\), \((l_{2}', m_{2}', n_{2}')\) and \((l_{3}', m_{3}', n_{3}')\) are the direction cosines of the three
principal stress directions with respect to the three reinforcement directions. The
\(z'\) direction has been assumed to be unreinforced for the purpose of this discus-
sion. When a small increment of load is applied on this element cracks are formed.
Figure 4.11: Bi-Directionally Reinforced Concrete Element Subjected to General Loads Prior to Cracking.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The shear stresses on the crack plane are no longer negligible if there are unbalanced loads in the unreinforced direction. In the crack normal direction the strain softening spring is initiated.

As before the strains in each steel direction is monitored (figure 4.12). When the strains in any reinforced direction reach the uniaxial cracking strain level the strain-softening in the crack normal direction is released. The tension-stiffening in each reinforcing direction sustains the forces released from the crack normal direction (figure 4.13).

**Figure 4.12:** Bi-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Strain $\epsilon < \epsilon_{cr}$ in the Steel Direction.

An inherent assumption made here to compute the force release in each steel direction is that the contribution of the bond in each direction is the same ($\Delta \sigma_z' = \Delta \sigma_x' = \Delta \sigma_y'$).
Figure 4.13: Bi-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Atleast One Strain $\epsilon \geq \epsilon_{cr}$ in the Steel Direction.
The increase in the stresses in each reinforced direction is:

$$\Delta \sigma'_x = \Delta \sigma'_y = \Delta \sigma'_n =$$

$$\Delta \sigma'_n \left\{ \frac{(l'_2 n'_3 - l'_1 n'_2)(m'_2 n'_3 - m'_3 n'_2) - (n'_1 m'_3 - m'_1 n'_3)(l'_2 n'_3 - n'_1 l'_3)}{n'_2 (l'_2 n'_3 - n'_2 l'_3) - l'_1 (m'_2 n'_3 - m'_3 n'_2)} \right\} \quad (4.22)$$

### 4.3.2.3 Uni-Directionally Reinforced Element

Consider next an element that has reinforcement only in one direction. When such an element is subjected to a general state of loads and it cracks, shear stresses develop very rapidly on the crack surface as this is a case of extreme anisotropy. Considering the element to be reinforced only in the $x'$ direction the equilibrium equations at incipient cracking in local steel direction ($x', y', z'$) are (figure 4.14):

$$\sigma_{x'} Al'_1 + \rho_{x'} \sigma_{x'} Al'_1 + \tau_{x'y'} Am'_1 + \tau_{x'z'} An'_1 = \sigma_{nn} Al'_1 \quad (4.23)$$

$$\sigma_{y'} Am'_1 + \tau_{x'y'} Al'_1 + \tau_{y'z'} An'_1 = \sigma_{nn} Am'_1 \quad (4.24)$$

$$\sigma_{z'} An'_1 + \tau_{y'z'} Am'_1 + \tau_{z'x'} Al'_1 = \sigma_{nn} An'_1 \quad (4.25)$$

With a small increment of load, cracks are formed and shear stresses begin to develop on the crack surface. Figure 4.15 shows the state of stress at this stage.

The strain in the steel direction are monitored till it reaches the uniaxial cracking strain, before releasing the strain-softening response in the crack normal direction. Immediately after the strain softening is released, the unreinforced direction becomes weak and shear stresses grow very rapidly on the crack plane to equilibrate the applied forces in the unreinforced direction (figure 4.16). The increase in stress along the steel direction obtained from equilibrium of forces on the crack plane is expressed as:

$$\Delta \sigma'_x = \Delta \sigma'_n =$$

$$\Delta \sigma'_n \left\{ \frac{(m'_2 n'_3 - m'_3 n'_2)(m'_2 l'_3 - m'_3 l'_2) - (l'_1 m'_3 - m'_1 n'_3)(n'_2 m'_3 - n'_3 m'_2)}{l'_1 m'_2 (n'_2 m'_3 - m'_2 n'_3) - l'_2 m'_1 (n'_2 m'_3 - m'_2 n'_3)} \right\} \quad (4.26)$$
Figure 4.14: Uni-Directionally Reinforced Concrete Element Subjected to General Loads Prior to Cracking.
Figure 4.15: Uni-Directionally Reinforced Concrete Element Subjected to General Loads After Cracking, With Strain \( \varepsilon < \varepsilon_{cr} \) in the Steel Direction.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 4.16: Uni-Directionally Reinforced Concrete Element Subjected to General Loads, With Strain $\epsilon \geq \epsilon_c$ in the Steel Direction.
Thus the bond released to the reinforcement in such specimen is a function of the orientation of the crack with respect to the reinforcement. It should be pointed out that if the crack is inclined to the reinforcement beyond a certain threshold angle there will be some unbalanced shear forces left on the crack plane and equilibrium is unattainable through straining of the reinforcement. Amongst the panels tested by Bhide and Collins (1986), panel PB18 (results of which are presented in chapter 6) shows rapid failure immediately after the initial cracks form.

The response of such specimen is dependent on not only the bond but also the shear stiffness mobilized on the crack plane. As the loading increases the shear stresses on the crack surface grow rapidly. Additional cracking is possible if the stresses build up to the cracking state again. These subsequent cracks are treated as brittle and the previously established tension stiffening spring are not altered. If such cracks form oriented beyond an angle of 45° with all the reinforcing direction(s) these cracks are treated as strain-softening cracks in plain concrete.

4.3.3 Compression Softening of Cracked Reinforced Concrete

Experimental investigations (Vecchio and Collins 1982, Kolleger 1988, and Dyngland 1989) have indicated that after cracking there is a significant reduction in the concrete compressive strength and stiffness parallel to the cracks. This is especially true when the amount of reinforcement is high and consequently crushing of concrete precedes yielding of reinforcement. Some of the factors affecting the behavior of cracked compressive struts are (1) cracking pattern (2) crack spacing (3) crack width (4) reinforcement direction with respect to the crack plane and (5) the amount of reinforcement. Spacing of cracks and the irregularity in their shapes may lead to eccentric loads and stress concentrations in the compressive strut, thus reducing its strength. Additionally, a skewed orientation of the reinforcement with respect to the crack direction may result in 'tearing' out of the bar at the crack surface and formation of secondary cracking. This causes severe local deterioration of the concrete.
Based on the observations in their tests, Vecchio and Collins (1982, 1986) have proposed an expression relating compressive strength of cracked concrete $\sigma_{\text{max}}^{\text{co}}$ to average tensile strains $\varepsilon''$ normal to the compressive strut as (figure 4.17):

$$\sigma_{\text{max}}^{\text{co}} = \frac{f'_c}{0.80 - 0.34\varepsilon''_c} \leq f'_c$$  \hspace{1cm} (4.27)

The uniaxial compressive strength of concrete is $f'_c$ and the corresponding compressive strain is $\varepsilon'_c$. The stress-strain relation in the strut is given by:

$$\sigma'_v = \sigma_{\text{max}}^{\text{co}} [2\{\varepsilon''_c - \varepsilon''_v\}^2]$$  \hspace{1cm} (4.28)

$\varepsilon''$ and $\varepsilon'''$ are the cracked concrete tensile and compressive strains, respectively. In this study equation 4.28 has been adopted to compute the stiffness of ‘intact’ concrete $E_{\text{co}}$ after cracking. The modulus of elasticity is continuously updated as:

$$E_{\text{co}} = \frac{\sigma'_v}{\varepsilon''_v}$$  \hspace{1cm} (4.29)

Figure 4.17: Compression-Softening in Cracked Reinforced Concrete, Vecchio and Collins (1982)
During analysis, the cracked concrete tensile strain ($\varepsilon''_t$) and compressive strain ($\varepsilon''_c$) are replaced by the maximum and minimum total principal strains, respectively. It should be pointed out that equation 4.28 was developed by (Vecchio and Collins 1982, 1986) for a particular crack orientation, while in this study it has been employed to compute the stiffness of the intact concrete ($E_{co}$) in any direction. But, as discussed in section 4.3.1, there is a rapid loss of stiffness in the crack normal direction due to the strain softening effect. Thus, the overall effect of using equation 4.28 is one of softening only the cracked concrete compressive strut, as proposed by Vecchio and Collins (1982,1986).

While most investigators accept the reduction in compressive strength for fact, there have been questions as to the level of this reduction. Both Kolleger (1988) and Dyngland (1989) have conducted tests on RC panels and concluded that equation 4.28 overestimates the strength reduction. In fact, they propose an upper limit of 20% to this reduction.

In this study another method to account for reduction in compressive strength of cracked concrete has been proposed. Upon cracking, the uniaxial compressive strength of concrete $f'_c$ has been reduced linearly as:

$$f_c = f'_c (1.0 - 0.2 \frac{\varepsilon_{max}}{\varepsilon'_c}) \geq 0.8f'_c$$  \hspace{1cm} (4.30)

where $\varepsilon_{max}$ is the current maximum principal tensile strain in concrete. $f_c$ is the modified uniaxial compressive stress in concrete and $\varepsilon'_c$ is the uniaxial compressive strain in concrete at peak compressive stress. Ottosen's triaxial strength envelope is modified to account for the change in the concrete compressive strength (figure 4.18). The corresponding strain at uniaxial compressive strength is also modified to account for the damage, due to cracking, and is expressed as:

$$\varepsilon_c = \varepsilon'_c (1.0 + 0.1 \frac{\varepsilon_{max}}{\varepsilon'_c})$$  \hspace{1cm} (4.31)

Numerical studies of test specimens using each of the above formulations for compressive strength of cracked concrete are presented in Chapter 6.
Figure 4.18: Compression-Softening in Cracked Reinforced Concrete, Present Study
4.3.4 Crack Interface Shear Transfer

Crack interface shear stiffness has been modeled in many studies by retaining a fixed fraction of the uncracked concrete shear stiffness as the crack shear stiffness in the crack stiffness matrix (equation 4.1). The shear stiffness of plain concrete elements is reduced to 1% of its value prior to cracking in this study. In RC elements, a small fraction of the uncracked concrete shear stiffness (20 to 50%) has been found to be sufficient, when these elements are reinforced in at least two orthogonal directions. In uni-directionally reinforced elements, (e.g. Bhide and Collins 1987) such an approximation of the shear stiffness leads to a very stiff response and a more sophisticated formulation to describe this phenomenon is necessary.

Experimental evidence (Millard and Johnson 1984, 1985; Vintzeleou and Tassios 1987) indicates that shear slip is affected not only by the corresponding shear stress but also by the associated crack opening. Likewise, the crack normal stress is affected by the crack opening and the shear stress, especially in reinforced concrete where the reinforcement constrains the crack opening. Attempts to include this coupling effect in analyses have met with difficulty, as these coupling components have been found to be unequal rendering the matrix indefinite and even ill-conditioned (Yoshikawa 1989). Therefore, in the present study an attempt has been made to include the coupling effects implicitly in the interface shear stiffness, using experimental data.

Millard and Johnson (1984, 1985) have conducted tests on aggregate interlock specimen while considering both reinforced concrete and externally held cracked concrete specimen. They have concluded that the initial axial stiffness normal to the plane of the crack which restrains crack widening is up to five times higher when specimen are reinforced in these tests because of local bond. However the stiffness diminishes towards that of an unbonded bar as the tension stiffening deteriorates at higher loads.

Similar studies have been conducted by Tassios and Vintzeleou (1987). In their tests on aggregate interlock behavior, pre-cracked concrete specimen were held against each other at constant stress normal to the crack plane and then sheared in
a direction parallel to the crack plane (figure 4.19). Vintzeleou and Tassios (1987) have also conducted dowel tests of reinforcement bars across frictionless concrete interfaces without any lateral confinement. These tests correspond to the externally held specimen tested by Millard and Johnson.

![Figure 4.19: Experimental Setup Used to Study Crack Interface Shear Load vs. Displacement Behavior Under Lateral Confinement ($\sigma_c$), Tassios and Vintzeleou (1987).](image)

In this study the results of the Tassios-Vintzeleou tests, (figure 4.20) representing crack interface shear stress-shear strain under different confining stresses, have been used to obtain the crack secant shear stiffness of cracked reinforced concrete elements. The bond stress is taken as a measure of the confinement imposed by the reinforcement across cracks. For any given crack, the crack shear strain and the crack normal bond stress are used to obtain the crack shear stiffness $G_{TV}$. As the bond between steel and concrete (tension-stiffening) is lost due to straining the crack shear stiffness decreases (aggregate interlock decreases) to that obtained from
the dowel tests, i.e., no confinement ( $\sigma_c = 0.0 \text{ MPa} $ ). It is expected that after yielding the confinement at the crack surface will be significantly reduced. But, the experimental data employed in this study is only for unyielded dowels. Therefore, the shear stiffness obtained from the experimental data is reduced using a relative stiffness factor $\delta$ which represents the ratio of the current cracked reinforced concrete stiffness normal to the crack to its value at cracking. The interface shear stiffness used in equation 4.1 is then given by:

$$ G_{cr} = \delta \ G_{TV} $$

(4.32)

![Image of Crack Interface Shear Stress-Strain Curves](image)

Figure 4.20: Crack Interface Shear Stress-Strain Curves at Various Levels of Confining Stresses Due to Aggregate Interlock and Dowel Action, Tassios and Vintzeleou (1987).

For numerical reasons, the minimum crack shear stiffness retained is 1% of the shear stiffness calculated for intact concrete at cracking.
Chapter 5

Numerical Implementation Aspects

5.1 Introduction

The three dimensional constitutive model for Reinforced concrete discussed in Chapter 3 and 4 has been implemented into a special purpose finite element program. For this purpose a finite element program used for studying metal forming processes, Foroozesh (1989), has been used as a base program for implementing the constitutive model for RC, layering procedures, numerical (secant) algorithm and the material point convergence procedures employed. For pre- and post-processing of the results the program is interfaced with the 'CAEDS', finite element graphics package available on the IBM 3090 system. The details of these procedures are described in this chapter.

5.2 Simulation of Concrete

In this study, for the simulation of concrete, a twenty noded isoparametric solid element and a eight noded degenerate shell element (Ahmad et al. 1970) have been employed. The solid element has been used primarily for ratifying the implementation of the constitutive model for concrete (Details of the material model ratification have been presented in section 3.2.3). The eight noded degenerate shell element has six degrees of freedom at each node: three translational and three rotational (figure 5.1). The 3-D stress-strain relationships for concrete given by equation 3.6 is condensed by enforcing the shell constraint of no (shell) normal stresses ($\sigma_2 = 0.0$). Thus the stress-strain relations for concrete in the shell local
coordinates \((x', y', z')\) is given by:

\[
\begin{pmatrix}
\sigma_{x'} \\
\sigma_{y'} \\
\sigma_{z'} \\
\tau_{x'y'} \\
\tau_{y'z'} \\
\tau_{x'z'}
\end{pmatrix} = \frac{E_{co}}{1 - \nu_{co}^2} \begin{pmatrix}
1 & \nu_{co} & 0 & 0 & 0 & 0 \\
\nu_{co} & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 - \nu_{co} & 0 & 0 \\
0 & 0 & 0 & 0 & K_1 \frac{1 - \nu_{co}}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & K_2 \frac{1 - \nu_{co}}{2}
\end{pmatrix} \begin{pmatrix}
\varepsilon_{x'} \\
\varepsilon_{y'} \\
\varepsilon_{z'} \\
\gamma_{x'y'} \\
\gamma_{y'z'} \\
\gamma_{x'z'}
\end{pmatrix} (5.1)
\]

where \(E_{co}\) and \(\nu_{co}\) are the secant stiffness and Poisson's ratio of the material respectively. \(K_1\) and \(K_2\) are the transverse shear energy correction factors in the \(y'z'\) and \(x'z'\) planes respectively.

---

**Figure 5.1: Eight-Noded Degenerate Shell Element.**

In order to calculate the stiffness of the material accurately, a 3x3 full Gaussian integration is required in the plane of the element. However, the use of full
integration has been shown to yield stiff responses (Zienkiewicz et al. 1971). This phenomenon occurs due to the extreme ratios of the bending stiffness to the shear and membrane stiffness. Shear locking has been shown to be suppressed with the use of reduced integration (2x2) of the shear terms (Zienkiewicz et al. 1971). In curved shell applications however it has been shown that the locking is due to the high ratios of the membrane stiffness to bending stiffness (Parisch 1979). The use of selective or uniformly reduced integration eliminates this problem. In material nonlinear problems however, the use of selectively reduced integration introduces difficulties in choosing the most optimal points for computing the stresses and strains for the different (bending, shear, membrane) stiffness components.

The use of uniformly reduced integration on the other hand, results in rank deficiencies in the stiffness matrix and the possibility of activating zero energy modes. A number of stabilization procedures have been proposed (eg. Belytchko and Tsay 1983, Milford and Schnobrich 1984) to suppress this rank deficiency. In the present study the full 3x3 and fully reduced 2x2 Gaussian integration procedures in the plane of the element have been implemented. In the shell thickness direction numerical integration is carried out using one, two or three sample points, located at ‘Gauss points’ in the thickness direction.

5.3 Simulation of Reinforcement

The reinforcement is treated as a smeared membrane layer with uniaxial stiffness properties oriented along the direction of the steel bars. The thickness of the reinforcement layer within an element is determined such that the cross-section of the reinforcement in the given element volume corresponds to the actual steel cross-section within the same volume. In this study a three dimensional degenerate shell element is employed for simulating the steel. A 3x3x1 Gaussian numerical integration procedure is employed in computing the stiffness of the steel.
5.4 Simulation of Cracks

A smeared cracking procedure has been employed to simulate a crack in this work. Here, the cracked solid is considered as a continuous medium and the cracks are modeled by means of inelastic strains (the rheological representation of cracks has been discussed in chapter 4). With such a treatment the effect of each crack is spread over a portion of the element.

A gradual reduction of the tensile stresses transferred at a cracked point causes a redistribution of stresses in the region. As discussed in section 2.4.2.1, a common feature of models employing the smeared crack procedure is their lack of objectivity in simulating crack propagation with respect to refinement of the finite element mesh (Bazant and Cedolin 1979, 1983). This objectivity with respect to mesh refinement can be achieved by modifying the constitutive law and making the fracture energy of concrete depend on the mesh size by introducing a parameter called ‘crack band width’ (Bazant and Oh, 1983). In regular meshes this parameter can be determined intuitively, but for irregular meshes and cracks skewed to the element sides this is no longer possible. However, there are some limitations. First, there is an upper limit for the crack band width (Bazant and Oh, 1983). In the analysis of large structures this limit may be exceeded and the introduction of a mesh dependent strength limit becomes attractive (Bazant and Cedolin 1979, 1983). Secondly, Bazant (1984) and Bazant, Belytschko and Chang (1984) have stated there is a lower limit for the crack band width, which roughly equals about three times the size of the aggregate. They pointed out that this lower limit has to do with the fact that the present formulation of the crack model is based on the local continuum theory. This theory would be unable to produce a detailed resolution of the stress and strain fields within and near the strain-softening region. Therefore, the size of an element within a finite element mesh should be greater than three times the size of the aggregate.

Oliver (1989) has proposed a method of evaluating the band width at cracked integration point which has been shown to be objective with mesh refinement. In this study this procedure has been employed.
For the simulation of more than one crack at a given sample point (non-orthogonal cracking) it has been found the strength criteria for crack initiation by itself is insufficient. It is quite possible that a series of cracks can be initiated resulting in 'numerical cracking' at that point. Thus, in this study the strength criteria is augmented by a minimum threshold angle criterion, whereby a new crack is initiated only if the strength criterion is violated and the crack normal of the new crack is directed at a skew larger than the threshold angle with respect to the normals of any previously existing cracks. Because such a criterion may cause the tensile stresses to build up to levels greater than the cracking strength of concrete under uniaxial tension, a tension-cutoff procedure is employed limiting the tensile stresses at a sample point to the uniaxial cracking strength of concrete. Typically the threshold angle used ranges from 15 to 30 degrees. However, it has been observed (e.g. de Borst and Nauta 1985) that for some problems this range is not sufficient and angles as large as 60 degree have been employed.

5.5 Layering Procedures

In this study layering procedures have been employed to capture the variation of material properties and the propagation of cracks through the thickness of the structure. In this study, two formulations have been used: an implicit layering procedure (Figueras and Owen 1983) and an explicit layering procedure (Barzegar 1989).

5.5.1 Implicit Layering Procedure

In the implicit layering procedure each finite element of the structure is made up of layers (figure 5.2). The constitutive aspects of the material are considered only at the layer level and element response is obtained from equilibrium and compatibility considerations. At layer interfaces, continuity of both the transverse shear stresses and displacements is required. Assuming that the 'normals' to the middle surface of the shell remain straight but do not necessarily remain normal to the middle surface of the shell, the latter continuity is satisfied but the former is not. This assumption
makes the transverse shear strain constant through the element thickness, which is a coarse approximation to the actual variation, even for homogeneous cross-sections. For homogeneous cross-sections, the transverse shear stress distribution is commonly accepted to be a parabolic function of the thickness coordinate \( z \). Therefore, a correction factor \( K \) must be introduced inorder to approximate, on an average basis, the transverse shear strain energy. For homogeneous layers a correction factor \( K_1 = K_2 = 5/6 \) is used in computing the strain energy. In the case of heterogeneous layers this value is no longer accurate. This is a limitation since these shear stress components cannot be used in nonlinear material model parameter computations. Figueras and Owen (1983) have proposed a more accurate method for computing shear strain energy correction factors for heterogeneous cross sections and have also suggested a method of obtaining shear stress components that are more accurate. They have employed this procedure in studying the behavior of orthotropic composite plates and have employed a plasticity-based material model description for this purpose. A brief description of this method (Figueras and Owen, 1983) follows.

![Figure 5.2: Implicit Layering Procedure](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The basic assumptions used by them in the development of this formulation are those of cylindrical bending and traction free surfaces. For a plate of thickness ‘h’, setting \( z = h/2 \) at the top surface and \( z = -h/2 \) at the bottom surface the transverse shear stress in the xz plane (obtained from equilibrium considerations) is:

\[
\tau_{xz} = -\int_{-h/2}^{h/2} \frac{\partial g}{\partial z} \, dz = -\frac{Q_x}{R_1} g(z)
\]  

(5.2)

where:

\( Q_x \) is the shear force on the xz plane;

\( R_1 = \int_{-h/2}^{h/2} D_1(\bar{z}) \bar{z}^2 \, d\bar{z} \) is the flexural plate stiffness in the x direction;

\( \bar{z} \) is a variable coordinate through the thickness;

\( g(z) = -\int_{-h/2}^{z} D_1(\bar{z}) \bar{z} \, d\bar{z} \) is the shear stress distribution function.

Expressing the strain energy component (per unit mid-plane surface) in terms of \( R_1 \) and \( g(z) \):

\[
w_s = \frac{Q_x^2}{R_1^2} \int_{-h/2}^{h/2} \frac{g^2(z)}{G_{13}(\bar{z})} \, d\bar{z}
\]

(5.3)

where \( G_{13}(\bar{z}) \) is the variable shear modulus in the xz plane.

Similarly the shear strain energy under the constant shear strain assumption is:

\[
\bar{w}_s = \frac{Q_x^2}{hG_1} \text{ where } \bar{w}_s = \frac{Q_x^2}{hG_1}
\]

(5.4)

\[
h \bar{G}_1 = \int_{-h/2}^{h/2} G_{13}(\bar{z}) \, d\bar{z}
\]

The strain energy correction factor in the xz plane \( K_2 \) is obtained by equating these energies and is expressed as:

\[
K_2 = \frac{\bar{w}_s}{w_s} = R_1^2 \left[ h \bar{G}_1 \int_{-h/2}^{h/2} \frac{g^2(z)}{G_{13}(\bar{z})} \, d\bar{z} \right]^{-1}
\]

(5.5)

The shear correction factor \( K_1 \) in the yz plane is obtained in the same way. In addition they have proposed expressions for computing the transverse shear stresses that account for material inhomogeneity. In the xz plane \( \tau_{xz} \) is obtained as:

\[
\tau_{xz}^{COR} = G_{13}(\bar{z}) \frac{g(z)}{\bar{g}}
\]

(5.6)

where:

\[
\bar{g} = \frac{1}{h} \int_{-h/2}^{h/2} g(z) \, d\bar{z}
\]

(5.7)
With this procedure an improved shear stress distribution through the thickness is obtained. The shear stress in the yz plane is computed in the same way. For numerical implementation all integrations are performed over the layer thickness and then summed up to account for material inhomogeneity.

The strain energy correction factor $K_1$ and $K_2$ represent the effect of the shear stress distribution, and also accounts for the material inhomogeneity, in an average sense over the entire cross-section of the element. However, in the present study a layerwise numerical integration procedure is being employed. Thus, in the present study the shear stress distribution function given by equation 5.6 is used. To approximately account for the stress distribution within each layer, a factor of $5/6$ for homogeneous layers, has been augmented to the expression given in equation 5.6.

### 5.5.2 Explicit Layering Procedure

The explicit layering scheme proposed by Barzegar (1989) employs stacked elements for considering variation of material properties through the thickness. Each element is associated with explicit degrees of freedom, some of which are constrained. In this procedure the number of active degrees in a node stack is large. The displacement degrees of freedom in the lower 'layers' (elements) are explicitly constrained in terms of the degrees of freedom of the layer above as:

$$U_2 = U_1 = \phi_{1y} \frac{h_1}{2} + \phi_{2y} \frac{h_2}{2}$$

$$V_2 = V_1 = \phi_{1z} \frac{h_1}{2} + \phi_{2z} \frac{h_2}{2}$$

$$W_1 = W_2$$

where $h_1$ and $h_2$ are the thicknesses of adjacent 'layers' (elements) in a stack and $U_1$, $V_1$, $W_1$, $\phi_{1x}$, $\phi_{1y}$, $U_2$, $V_2$, $W_2$ and $\phi_{2x}$, $\phi_{2y}$ are the translational and rotational degrees of freedom of the corresponding nodes of adjacent elements in the stack. Figure 5.3 shows the details of this procedure. The primary advantage of this layering procedure is in its ability to capture the variation of the transverse shear strain and stress through the thickness accurately. However, computationally this...
procedure is expensive because of the increase in nodal degrees of freedom. The constraint equations generated by this layering procedure are implemented via a lagrange multiplier technique described in Cook(1981).

In treating reinforcement, some modification of the explicit layering procedure has been made in this study. The steel layer along with the concrete layer adjacent to it are grouped into a single element through the use of the implicit layering procedure.

5.6 Computational Scheme

The total secant finite element formulation used in this study requires the solution of the following equation for the displacements.
Here \( \{K\} \) is the global secant stiffness matrix, \([F]\) is the total load vector and \([U]\) is the total displacement vector. Details of formulating the stiffness matrix and load vector are presented in Bathe (1978). For nonlinear material problems the stiffness matrix \( \{K\} \) depends on the stresses within the material and so an iterative solution procedure is necessary. Figure 5.4 illustrates the functioning of the secant algorithm and the path to convergence. The system of equations obtained in equation 5.11 are solved by the Gauss elimination method using the active column storage technique (Bathe 1982).

\[
\{K\} [U] = [F] \tag{5.11}
\]

5.6.1 Convergence Procedures

For a solution strategy based on iterative methods to be effective a realistic criteria should be employed for the termination of the iteration at a given load.
level. At the end of each iteration, the solution obtained should be checked to see if it has converged within the preset tolerances (3-5%) or if divergence was detected. If convergence tolerances are not strict, inaccurate results may be obtained, and if the tolerances are too strict, much computational effort is spent on needless accuracy.

The convergence criterion employed depends on the type of problem being examined. For example when the displaced configuration is being sought at a given load level, convergence is assumed if the displacement at the end of each iteration is within a specified tolerance of the displacement at the end of the previous iteration. However, in some problems the variation in the displacement may be very small within each iteration while the actual solution may be far from the one obtained using this procedure. In such cases a convergence criterion based on requiring the out of balance load vector at any iteration to be within a tolerance of the load increment may be more appropriate. In order to provide some indication of when both the displacement and the forces are near their equilibrium position a criterion based on energy is sometimes employed. Here convergence is assumed when the increment in internal energy (work done by the out of balance load vector in the previous iteration) is within a prescribed tolerance of the initial internal energy increment computed in the first iteration of the current load increment.

For the total secant stiffness formulation implemented in the present study an energy based criterion enforced at each integration point of each layer within each element is employed (Barzegar 1986). The details of its implementation in the present study is discussed in this section.

5.6.1.1 Intact concrete Response

Pre Peak Convergence
The displacements obtained from solution of equation 5.11 are processed at each sampling point to obtain the total strain components \( \{\epsilon\}_{x,y,z} \) at these points. The total stresses \( \{\sigma\}_{x,y,z} \) at these points are recovered using equation 4.9. The principal stresses are then computed. The principal strains in the uncracked concrete are computed using the principal stresses. If all these strain components are less than
the corresponding largest strain components previously registered at this point the point is unloading and convergence is assumed. If any strain component is found to be greater than its previously stored value, loading is assumed and the convergence criterion described here is enforced. Figure 5.5 illustrates the convergence procedure employed in the pre-peak region at a Gauss point under a uniaxial state of stress. $E_1$ is the used secant stiffness for concrete in the current iteration resulting in a concrete strain $\epsilon_1$. Using the calculated stress $\sigma_1 = E_1 \epsilon_1$ a new stiffness $E_1'$ is determined. If the difference between the used stiffness $E_1$ and the new stiffness $E_1'$ is less than the tolerance (typically 3-5%), convergence of the stress-strain law is assumed. If this requirement is not satisfied then a new stiffness, $E_2$, is obtained as:

$$E_2 = \text{Minimum}(\frac{E_1 + E_1'}{2}, E_1(1 - \text{tolerance}))$$

(5.12)

As seen in figure 5.5 the convergence path is one of gradual softening at the Gauss point leading to release of stresses in the region.

Figure 5.5: Convergence Path at a Point in Concrete Prior to Crushing.
The convergence procedure employed to modify the Poisson’s ratio is similar to the one employed for concrete stiffness. When the used Poisson’s ratio \(\nu_1\) is less than the updated one \(\nu_2\) by more than the specified tolerance, a new Poisson’s ratio obtained as:

\[
\nu_3 = \nu_1 (1.0 + \text{tolerance})
\]  

To detect limit points in the stress-strain curve the tolerances used are automatically reduced internally. Further, overshooting of the stresses at the Gauss point is also corrected by a gradual reduction of the stiffness.

**Post-Peak Convergence**

In the post-peak strain-softening zone similar modification of the secant modulus and Poisson’s ratio are made (figure 5.6). If the used stiffness \(E_1\) is less than its current value, \(E'_1\), by more than the specified tolerance then a new stiffness is required. The new secant stiffness employed is \(E_2 = E_1(1 - \text{tolerance})\). As mentioned in section 3.3 the Poisson’s ratio corresponding to \(E_2\) is obtained as \(\nu_2\) such that the bulk modulus at peak compressive stress remains unchanged. The new Poisson’s ratio is then calculated as \(\nu_2 = 1.005\nu_2\).

### 5.6.1.2 Crack Interface Response

At each sampling point the total global strain components \(\{\varepsilon\}_{xyz}\) are recovered from the nodal displacements. These strain components and the cracked concrete stiffness (equation 4.7) are used to obtain the total global stresses \(\{\sigma\}_{xyz}\) at that sampling point. The crack stress components of each crack \(\{\sigma\}_{n,t}\) are then obtained from the global stresses using the stress transformation matrix.

The current crack normal strain \(\varepsilon_1\) is obtained from the crack normal stress \(\sigma_{nn}\) and the employed crack stiffness \(E_1\) (figure 5.7). If this strain component is less than its largest value previously registered at this point the point is unloading and convergence is assumed. If the strain is found to be greater than the previously stored strain loading is assumed and a convergence criterion described here is enforced. The new stiffness \(E_2\) is obtained from the crack strain \(\varepsilon_1\) and the stress-strain relationship employed for the crack. The total area under the stress-strain
diagram for the crack is $A$, while the area under the stress-strain curve between $E_1$ and $E_2$ is $A_1$. Convergence is assumed if the area $A_1$ in the current iteration satisfies the relationship:

$$A_1 \leq A \times \text{tolerance}$$

where the tolerance is typically 3-5%. If this not satisfied $E_2$ is assumed to be the new stiffness and further iterations are carried out.

Great care is needed in employing the above criterion in the initial stages of an opening crack (figure 5.8). The change in the stiffness over successive iterations is so small that equation 5.14 would indicate convergence. However, the stress point has violated the crack strength $f_{ct}$ as seen in figure 5.8. In this situation the convergence is rejected and a new stiffness $E'_2$ is employed. This stiffness $E'_2$ is obtained such that the area $A_2$ under the stress strain curve between $E_1$ and $E'_2$ is obtained as :

$$A_2 = \text{tolerance} \times A$$
Figure 5.7: Convergence Path in a Strain Softening Cracked Point.

More iterations are carried out until the stress point satisfies the constitutive law at the point.

5.6.1.3 Tension-stiffening Response

The tension-stiffening strain $\epsilon_1$ is obtained by transforming the global strains $\{\epsilon\}_{e\nu}$ through the strain transformation matrix. If this strain component is less than its largest strain level previously registered at this point the point is unloading and convergence is assumed. If the strain is found to be greater than the previously stored strain, loading is assumed, and a convergence criterion described here is enforced. The tension-stiffening strain $\epsilon_1$ is employed to compute the new stiffness $E_2$ using the stress-strain relationship (figure 5.9). The total area under the stress-strain diagram is $A$. The area under the stress-strain curve between $E_1$, the stiffness used in the current iteration, and $E_2$ is $A_1$. Convergence is assumed if the following
relationship is satisfied.

\[ A_1 \leq \text{tolerance} \times A \]  \hspace{1cm} (5.16)

If this criterion is violated then the new stiffness \( E_2 \) is employed and more iterations are required.

In the initial stages the tension stiffness is relatively high and the above criterion may be easily satisfied. However, the stress point may overshoots the cracking strength as shown in figure 5.10. In this case the convergence is rejected and a new stiffness \( E_3 \) should be employed. \( E_3 \) is obtained such that the area \( A_2 \) enclosed between \( E_1 \) and \( E_3 \) is given by:

\[ A_2 = \text{tolerance} \times A \]  \hspace{1cm} (5.17)
5.6.1.4 Reinforcement Response

The global strain components obtained from displacements are transformed to the steel fibre orientation $\varepsilon_n$. If this strain component is less than the largest strain previously registered at this point the point is unloading and convergence is assumed. If the strain is found to be greater than the previously stored strain loading is assumed and a convergence criterion described here is enforced. Figure 5.11 shows the convergence procedure employed in the post-yield zone in steel. The stress in steel is obtained from the relationship $\sigma_s = E_1\varepsilon_n$ where $E_1$ is the secant stiffness employed in the current iteration. The new stiffness $E_2$ is obtained from $\varepsilon_n$ and the stress-strain diagram used for steel. If $A$ is the total area under the stress-strain diagram for steel and $A_1$ is the area under the stress-strain diagram between $E_1$ and $E_2$ then convergence at the given sample point is assumed if the following criterion is satisfied:

$$A_1 \leq A \times \text{tolerance} \quad (5.18)$$
Figure 5.10: Detection and Correction for Overshooting at a Tension-Stiffening Point.
If the stress point $\sigma_s$ overshoots the ultimate stress $\sigma_u$ then this convergence is rejected (figure 5.12). The new stiffness $E'_2$ is obtained such that the area $A_2$ between $E_1$ and $E'_2$ meets the following requirement:

$$A_2 \leq A \ast \text{ tolerance}$$  \hspace{1cm} (5.19)

**5.6.2 Numerical Algorithm**

The complete listing of the program along with sample input files are given in the appendix. A brief outline of the steps employed in the nonlinear algorithm are presented here.

1. For any given increment, the converged stiffness $\{K\}$ from the previous load increment and the current load vector $[F]$ are used in equation 5.11 to solve for the displacements $[U]$. 

![Figure 5.11: Convergence Path in Steel in the Post-Yielded Zone.](image)
2. From these displacements the total strains and total stresses are recovered at each sample point within each layer of each element.

3. Checks are made to determine the loading index at each point, within each rheological component based on the current strain and previously stored strain levels within each respective component. If this index indicates unloading, convergence is assumed.

4. If the loading index indicates the point is loading, the convergence criterion is employed to determine if the stiffness component is to be updated. If this is necessary then the convergence flag is set on and the stiffness matrix (equation 4.6) is reformulated at that point.

5. If each sample point within each layer of each element has been processed continue to step 6 else return to step 2.
6. If the convergence flag is on, update the stiffness matrix \( \{K\} \) and return to step 1 provided the maximum number of iterations are not exceeded. If the iteration count exceeds the maximum allowable iterations (typically 50), go to step 8.

7. If convergence requirements have been satisfied, the results are printed. Return to step 1 with the next load increment. If all load increments have been applied stop the process.

8. If iterations have been exceeded the unconverged results are printed and the process is stopped.

5.6.3 Load Increment Size

To properly and efficiently capture the nonlinear behavior of reinforced concrete, it is necessary to keep the load increment size reasonably small. This provides a more accurate representation of the material behavior and requires fewer iterations to converge at each load step. Large load steps may force nonlinear behavior in some regions of the structure that may have remained elastic due to the softening in other parts of the structure. Load increments of 1 to 3% of the ultimate load have been used in this study.
Chapter 6

Numerical Studies and Discussion of Results

6.1 Introduction

To demonstrate the capabilities of the analytical model for reinforced concrete proposed in this study a number of RC test specimen under the action of various loads have been examined. The details of these studies are reported in this chapter.

First, the material model parameters required to establish the shape of the tension-stiffening spring, described in section 4.3, have been evaluated by analyzing a number of test specimen subjected to uniaxial tension. Panel elements under the action of uniform membrane stress fields have been examined to highlight the influence of the different components of the analytical model in capturing the dominant response behavior.

Wall elements under the action of vertical and horizontal in-plane loads, applied proportionally and non-proportionally, have then been analyzed. Beam and slab components have also been analyzed to evaluate the model capabilities under the action of flexural and torsional loads. The load-deformation response, cracking patterns, stress and strain contours in concrete and steel have been examined to determine the dominant response characteristics and modes of failure of these specimen. Appropriate comparisons with results from other investigations and the ACI code provisions have also been included.
6.2 RC Elements Subjected to In-plane Loads

6.2.1 Membrane Loading Without Stress Gradients

6.2.1.1 Uniaxial Loadings

Uniaxial tension test specimen were selected for this study primarily to establish the post cracking stress-strain response curves of reinforced concrete employed in the analysis. The criterion used for selecting test specimen included the shape of RC element, the percentage of reinforcement, the orientation of reinforcement and the material properties of concrete and steel. To consider the influence of this broad range of parameters the examples were selected from Rizkalla and Hwang (1984), Shima et al. (1987), and Bhide and Collins (1987).

The parameters defining the shape of the tension-stiffening curve shown in figure 4.4, in section 4.3.1, were obtained by calibrating them to the uniaxial test results discussed here. The values of $\alpha$, $\beta$, $\gamma$ and $\delta$ (figure 4.4) obtained were $0.5\epsilon_y$, $\epsilon_y$, $0.35f_{cr}$, and $0.10f_{cr}$, respectively, where $\epsilon_y$ is the strain in steel at yield and $f_{cr}$ is the cracking stress in concrete.

The dimensions and material properties of the test specimens is presented in table 6.1. Only a single shell element is used to model the specimen since the stress field is uniform. The cross-section of the RC specimen were modelled using the implicit layering procedure described in section 5.5.3. A single layer is used to model the concrete, while steel in each different direction is treated as an individual layer with the appropriate layer thickness. A $3\times3\times1$ Gaussian numerical integration procedure has been employed to compute the stiffnesses of both concrete and steel. Figure 6.1 shows the tensile stress-strain behavior of a reinforced concrete bar Shima #3 (specimen number), tested by Shima et al. (1987). The load deformation response of this bar is shown in figure 6.2. The analytical results obtained in the present study are also presented in these figures and are labeled as FEM1. Both the axial load-strain as well as the concrete tensile stress-strain response characteristics appear to be well represented by the analytical prediction up to ultimate. The oscillations in the tensile stress-strain response in the initial stages of the analysis is
due to excessive reduction of the concrete stiffness and corrects itself with additional loading. The mode of failure was yielding of the steel. The ultimate load as reported in table 6.2 compares well with the experimental results.

Table 6.1: Dimensions and Material Properties of Uniaxial Tension Specimen

<table>
<thead>
<tr>
<th>Speci. Label</th>
<th>Dimensions (mm)</th>
<th>Concrete</th>
<th>Reinforcement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>f'_c (MPa)</td>
<td>e'_c (mm)</td>
<td>f'_t (MPa)</td>
</tr>
<tr>
<td>Shima #3</td>
<td>2700x150x200</td>
<td>25.7</td>
<td>0.002</td>
</tr>
<tr>
<td>Shima #5</td>
<td>2700x250x250</td>
<td>25</td>
<td>0.002</td>
</tr>
<tr>
<td>Riskalla #2</td>
<td>762x305x178</td>
<td>34.5</td>
<td>0.002</td>
</tr>
<tr>
<td>PB13</td>
<td>890x890x70</td>
<td>23.4</td>
<td>0.002</td>
</tr>
</tbody>
</table>

Note: Initial Stiffness in Concrete is $\frac{f'_c}{e'_c}$.

Table 6.2: Results of Uniaxial Tension Tests on RC Specimen

<table>
<thead>
<tr>
<th>Specimen</th>
<th>Finite Element Analysis</th>
<th>Experiment</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ultimate Load (MPa)</td>
<td>Failure Mode</td>
<td>Ultimate Load (MPa)</td>
</tr>
<tr>
<td>Shima #3</td>
<td>6.10</td>
<td>Yielding</td>
<td>6.01</td>
</tr>
<tr>
<td>Shima #5</td>
<td>5.08</td>
<td>Yielding</td>
<td>4.82</td>
</tr>
<tr>
<td>Riskalla #2</td>
<td>7.1</td>
<td>Yielding</td>
<td>6.55</td>
</tr>
<tr>
<td>PB13</td>
<td>4.76</td>
<td>Yielding</td>
<td>4.70</td>
</tr>
</tbody>
</table>

Specimen Shima #5 (specimen number), tested by Shima et al. (1987), has also been analysed in this study. This specimen has a higher percentage of steel of 2%, compared to 1% in specimen Shima #3. The ultimate load obtained from the analyses and the experiment is reported in table 6.2. As expected, the increase in reinforcement percentage is reflected in a higher load carrying capacity. The tension-stiffening formulation employed in the analysis is not implicitly dependent on the percentage of reinforcement present in the specimen. However, the analytical...
Figure 6.1: Tensile Stress-Strain Response, Specimen Shima #3.

Figure 6.2: Axial Load-Deformation Response, Specimen Shima #3.
Figure 6.3: Tensile Stress-Strain Response, Specimen Shima #5.

Figure 6.4: Axial Load-Deformation Response, Specimen Shima #5.
prediction of the tensile stress-strain response and the load deformation response is very close to the experimental result (figure 6.3 and 6.4). Thus, the use of a smeared representation of the reinforcement and tension stiffening for a single bar appears to be quite effective in simulating the load deformation responses.

While the tests conducted by Shima et al. (1987) were on RC bars, those conducted by Rizkalla and Hwang (1984) and Bhide and Collins (1986) were done using RC panel elements. Figure 6.5 shows the tensile stress-strain response predictions for specimen Rizkalla #2 along with the experimental results from Rizkalla and Hwang (1984). The analytical predictions closely follow the experimental results up to failure. The axial load-deformation response for this specimen is shown in figure 6.6. The ultimate load is reported in table 6.2. Specimen PB13, tested by Bhide and Collins (1987), has also been analyzed in this study and the ultimate load is given in table 6.2. The tensile stress-strain response and the load deformation response for specimen PB13 is shown in figure 6.7 and 6.8 respectively.

The analytical model does not have any size dependent parameters built into it explicitly. However, the tensile stress-strain response obtained from the analytical model is quite satisfactory when compared with the experimental results. This would indicate that the size and shape of the specimen do not appear to have influenced the tensile stress-strain response and that employing a smeared tension-stiffening layer to represent the transfer of stresses through bond is reasonable.

6.2.1.2 General In-plane Loading

In the preceding section, the capabilities of the analytical model in predicting the response of uniaxially loaded tensile test specimen has been reported. In this section, its ability to predict the response of test specimen under uniform general membrane stress fields is examined. Unless otherwise stated a threshold angle of 15° has been used, for the initiation of a new crack with respect to previously existing cracks, in these analyses. A single shell element is used to describe the specimen, since only uniform stress fields are present in these specimen. A 3x3x1 Gaussian numerical integration procedure has been employed for computing the stiffnesses of
Figure 6.5: Tensile Stress-Strain Response, Specimen Rizkalla #2.

Figure 6.6: Axial Load-Deformation Response, Specimen Rizkalla #2.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6.7: Tensile Stress-Strain Response, Specimen PB13.

Figure 6.8: Axial Load-Deformation Response, Specimen PB13.
the steel and concrete layers. The cross-section of the RC specimen were modelled using the implicit layering procedure described in section 5.5.3. A single layer is used to model the concrete, while each set of steel bars in a direction is treated as a separate layer with the appropriate thickness.

**Panels S1, S3, S5, S6 and S7**

To study the influence of the orientation of the reinforcement and anisotropic arrangement of the reinforcement on the accuracy of prediction of the analytical model, a series of panel elements tested by Dyngland (1989) were selected in this study. This series of specimen consisted of three isotropically reinforced specimen (S1, S3 and S6) and two with anisotropic arrangement (S5, S7) of steel. The material properties, type of loading and geometry of the test specimen is given in tables 6.3 and 6.4. These specimen were all loaded in uni-directional tension. Figures 6.9, 6.10 and 6.12 show the applied load deformation response of isotropically reinforced specimen S1, S3 and S6 respectively. The analytical predictions of the deformational response appear to closely follow the experimental results in each case indicating the use of tension stiffening in direction(s) skewed to the loading direction (also the direction of initial cracks) is reasonable.

Table 6.3: Loading Procedures for Tests on Membrane Elements

<table>
<thead>
<tr>
<th>Specimen</th>
<th>Loading</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Uniaxial Tension</td>
</tr>
<tr>
<td>S3</td>
<td>Uniaxial Tension</td>
</tr>
<tr>
<td>S5</td>
<td>Uniaxial Tension</td>
</tr>
<tr>
<td>S6</td>
<td>Uniaxial Tension</td>
</tr>
<tr>
<td>S7</td>
<td>Uniaxial Tension</td>
</tr>
<tr>
<td>PB18</td>
<td>Pure-Shear</td>
</tr>
<tr>
<td>PB19</td>
<td>Tension:Shear :: 1.01:1</td>
</tr>
<tr>
<td>PB20</td>
<td>Tension:Shear :: 2.04:1</td>
</tr>
<tr>
<td>PB21</td>
<td>Tension:Shear :: 3.08:1</td>
</tr>
<tr>
<td>PB22</td>
<td>Tension:Shear :: 6.09:1</td>
</tr>
</tbody>
</table>

Analytical predictions of the response of specimen S5, shown in figure 6.11, indicates a good match with the experimental results of Dyngland (1989). As expected,
Table 6.4: Dimensions and Material Properties of Membrane Elements

<table>
<thead>
<tr>
<th>Speci. Model</th>
<th>Dimensions (mm)</th>
<th>Concrete</th>
<th>Reinforcement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f'_c$ (MPa)</td>
<td>$\varepsilon'_c$ (MPa)</td>
<td>$f'_y$ (MPa)</td>
</tr>
<tr>
<td>$S1^1$</td>
<td>630x630x100</td>
<td>23.1</td>
<td>0.002</td>
</tr>
<tr>
<td>$S3^2$</td>
<td>630x630x100</td>
<td>21.8</td>
<td>0.002</td>
</tr>
<tr>
<td>$S5^2$</td>
<td>630x630x100</td>
<td>22.9</td>
<td>0.002</td>
</tr>
<tr>
<td>$S6^3$</td>
<td>630x630x100</td>
<td>24.3</td>
<td>0.002</td>
</tr>
<tr>
<td>$S7^3$</td>
<td>630x630x100</td>
<td>23.1</td>
<td>0.002</td>
</tr>
<tr>
<td>PB18^1</td>
<td>890x890x70</td>
<td>25.3</td>
<td>0.002</td>
</tr>
<tr>
<td>PB19^1</td>
<td>890x890x70</td>
<td>20.</td>
<td>0.002</td>
</tr>
<tr>
<td>PB20^1</td>
<td>890x890x70</td>
<td>21.7</td>
<td>0.002</td>
</tr>
<tr>
<td>PB21^1</td>
<td>890x890x70</td>
<td>21.8</td>
<td>0.002</td>
</tr>
<tr>
<td>PB22^1</td>
<td>890x890x70</td>
<td>17.6</td>
<td>0.002</td>
</tr>
</tbody>
</table>

1: Reinforcement mesh at 0° to loading direction
2: Reinforcement mesh at 45° to loading direction
3: Reinforcement mesh at 18.4° to loading direction

Concrete initial Stiffness $\frac{2f'_c}{\varepsilon'_c}$
the rotation of the principal directions towards the weakly reinforced direction is observed in the analysis. This indicates that the development of the shear stresses on the crack surface, that are responsible for the rotation of the principal directions, are well represented by the crack interface shear model employed in this study.

Figure 6.13 shows the analytical prediction for specimen S7 together with the experimental result. While the ultimate load predicted for this specimen is close to the experimental one, the deformational response predicted is more flexible than the experimental results after initial cracking due to 'global' yielding of the reinforcement in the weaker direction. One possible reason for the stiffer response observed in the experiment, is the resistance offered by the welds connecting the orthogonally reinforced bars (forming a mesh) to straining in the weak direction, which is not accounted for in the analysis. Thus, while the weaker direction yielded in the experiment the strains registered in that direction were reduced compared with those obtained in the analysis.
Figure 6.10: Axial Stress-Strain Response, Specimen S3

Figure 6.11: Axial Stress-Strain Response, Specimen S5
Figure 6.12: Axial Stress-Strain Response, Specimen S6

Figure 6.13: Axial Stress-Strain Response, Specimen S7

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Table 6.5: Results of Tests on Membrane Elements

<table>
<thead>
<tr>
<th>Specimen</th>
<th>Finite Element Analysis</th>
<th>Experiment</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ultimate Load (MPa)</td>
<td>Failure Mode</td>
<td>Ultimate Load (MPa)</td>
</tr>
<tr>
<td>S1</td>
<td>5.0</td>
<td>Yielding</td>
<td>4.78</td>
</tr>
<tr>
<td>S3</td>
<td>5.0</td>
<td>Yielding</td>
<td>4.84</td>
</tr>
<tr>
<td>S5</td>
<td>2.76</td>
<td>Yielding</td>
<td>3.18</td>
</tr>
<tr>
<td>S6</td>
<td>5.0</td>
<td>Yielding</td>
<td>4.84</td>
</tr>
<tr>
<td>S7</td>
<td>4.63</td>
<td>Yielding</td>
<td>4.57</td>
</tr>
<tr>
<td>PB18</td>
<td>1.87 (2.23)</td>
<td>Sliding</td>
<td>1.65</td>
</tr>
<tr>
<td>PB19</td>
<td>1.58 (2.05)</td>
<td>Sliding</td>
<td>1.23</td>
</tr>
<tr>
<td>PB20</td>
<td>1.26</td>
<td>Sliding</td>
<td>1.41</td>
</tr>
<tr>
<td>PB21</td>
<td>1.37</td>
<td>Sliding</td>
<td>1.38</td>
</tr>
<tr>
<td>PB22</td>
<td>0.97</td>
<td>Sliding</td>
<td>1.01</td>
</tr>
</tbody>
</table>

1 : Threshold Angle 10°
2 : Threshold Angle 15°

Panel PB21: Shear and Tension

Panel PB21, tested by Bhide and Collins (1987) is a uniaxially reinforced panel subjected to simultaneous shear and tension. The material properties, geometry and loading information are given in table 6.3. The initial crack formed (defined by the crack normal direction) at an 18° skew with the reinforcing direction. Due to the lack of reinforcement in one direction, a significant shearing stress is generated on the crack plane to equilibrate the external forces in the unreinforced direction. This causes the principal loading direction to rotate rapidly towards the unreinforced (weak) direction, leading eventually to further cracking in a new direction and failure. In the analysis, these secondary cracks form at an inclination of nearly 34° with respect to the reinforcement. The shear stress-strain response, shown in figure 6.14 indicates that the analytical prediction compares well with the experimental results. The tensile stress-strain response is shown in figure 6.18. The stress-strain response in the reinforcing direction, shown in figure 6.15, appears to be well represented by the analysis. The stress-strain response in the unreinforced direction.
(figure 6.16) shows that the analytical response is stiff compared to that observed in the experiment. While the initial response is controlled by the sliding behavior on the crack plane the formation of secondary cracks and the large straining in the unreinforced direction is the cause of 'failure'. This indicates that beyond a certain skew between the crack direction and the reinforcement the crack response is like that of a plain concrete specimen and the influence of the reinforcement and the tension-stiffening on the load carrying capacity is negligible. The load-maximum principal strain response is shown in figure 6.17. The variation in the direction of the principal stress and strain direction with applied loads is shown in figure 6.19 and 6.20 respectively. The analytical predictions of the rotations in the principal stress direction compares reasonably well with the experimental results. Thus, the crack interface shear stresses modelled by the aggregate interlock model, responsible for rotation of the principal stress directions, appears to be well represented.

The severe rotation of the principal strain direction is not well simulated by the analysis. The reason for this is the delay in the formation of secondary cracks imposed by the minimum threshold angle (15°) requirement as discussed in section 5.3 (simulation of cracks). The experimental results indicate a rapid rotation of the principal strain direction after initial cracking and formation of secondary cracks at a load of 1.07MPa. The formation of secondary cracks is initiated only at a load of 1.17MPa in the analysis. The ultimate load prediction and the experimentally obtained values, given in table 6.5, compare reasonably.

**Panel PB18: Pure Shear**

Panel PB18 is a uniaxially reinforced panel subjected to pure shear load. The material properties, geometry and loading information are given in table 6.3. The initial cracks formation is at 45° to the reinforcing direction. Due to the lack of reinforcement in one direction, a significant shearing stress is generated on the crack plane to balance the external forces in the unreinforced direction. This causes the principal loading direction to rotate rapidly towards the unreinforced direction, leading to further cracking and failure almost immediately after the first cracks are formed. This test conducted by Bhide and Collins (1987) is a case of extreme anisotropy. The shear stress-strain response is shown in figure 6.21. The analytical response is
Figure 6.14: Load vs. Strain Response, Specimen PB21

Figure 6.15: Load vs. Transverse Strain Response, Specimen PB21
Figure 6.16: Load vs. Longitudinal Strain Response, Specimen PB21

Figure 6.17: Load vs. Maximum Principal Strain Response, Specimen PB21
Figure 6.18: Tensile Stress-Strain Response, Specimen PB21

Figure 6.19: Principal Stress Direction vs. Axial Load, Specimen PB21
shown for two allowable threshold angles established for crack formation. When the threshold angle is 15° the secondary crack formation is delayed significantly, resulting in a very stiff load deformation response. When the threshold angle is reduced to 10° the ultimate load is reduced and the results are closer to that observed in the experiments. The tensile stress-strain response is shown in figure 6.23. The stress-strain response in the reinforcing direction, shown in figure 6.22, appears to be well represented by either analysis. This implies that beyond a certain skew between the cracks formed and the reinforcement the crack response is like that of a plain concrete specimen and the influence of the reinforcement on their response is negligible. The stress-strain response in the uncracked direction (figure 6.25) shows that the analytical response is stiff compared to that of the experiment. This indicates that while the initial response is controlled by the sliding behavior on the crack plane, the formation of secondary cracks and the large straining is the cause of 'failure'. The load-maximum principal strain response is shown in figure 6.24.

The load-principal stress and strain direction responses are shown in figure 6.26.
and 6.27 respectively. Results of other tests conducted in this series by Bhide and Collins (1987), specimen PB19 PB20 and PB22, are shown in table 6.5 along with the ultimate load prediction obtained in this study.

**Panel PV19: Pure Shear**

RC panel PV19 is an anisotropically reinforced panel element subjected to pure shear tested by Vecchio and Collins (1982). The reinforcement in the transverse direction is lower (0.78%) compared to the longitudinal direction (1.78%). After cracking, this anisotropy causes the principal stress and strain directions to rotate toward the weaker direction as seen in figure 6.33 and figure 6.34 respectively. The increased straining along the weaker direction causes the transverse steel direction to yield as observed in figure 6.30. The longitudinal strain component increases very slowly and remains in the elastic range at failure (figure 6.29). The tensile stress-strain response from the tests, shown in figure 6.31, clearly shows that some tensile stresses are sustained by the concrete, even after yielding of the transverse steel. This is also reproduced in the analysis. After cracking, a large amount of the load transfer capacity is generated in the weaker direction through the crack interface shear stresses. However, with the rapid increase in loads, this interface shear transfer capacity is exhausted and a shear sliding failure is predicted by the analysis FEM1 (figure 6.28). The compressive stress-strain behavior obtained from analysis FEM1 as seen in figure 6.32 clearly indicates that the capacity of the compressive strut is not exhausted. The analytical model FEM2 predicts crushing of the compressive strut due to the strain-based reduction of the uniaxial compressive strength (nearly 40% reduction). The ultimate load predicted by FEM1 is 4.0 MPa while FEM2 predicted failure at 3.55 MPa. The experimental failure load of 3.9MPa (table 6.8). More significantly the failure mode observed in the experiment was of failure along the shear plane as predicted by FEM1. Premature crushing observed in FEM2, was caused by excessive softening and strength reduction of the compressive strut.

**Panel PV27: Pure Shear**

RC panel element PV27 is an isotropically reinforced specimen (ρ = 1.75%) subjected to pure shear. The shear stress-strain behavior is shown in figure 6.35. It is
Figure 6.21: Load vs. Strain Response, Specimen PB18

Figure 6.22: Load vs. Transverse Strain Response, Specimen PB18
Figure 6.23: Load vs. Longitudinal Strain Response, Specimen PB18

Figure 6.24: Load vs. Maximum Principal Strain Response, Specimen PB18
Figure 6.25: Tensile Stress-Strain Response, Specimen PB18

Figure 6.26: Principal Stress Direction vs. Axial Load, Specimen PB18

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6.27: Principal Strain Direction vs. Axial Load, Specimen PB18

Table 6.6: Loading Procedures For Tests on Membrane Elements

<table>
<thead>
<tr>
<th>Specimen</th>
<th>Loading</th>
</tr>
</thead>
<tbody>
<tr>
<td>PV19</td>
<td>pure-shear</td>
</tr>
<tr>
<td>PV27</td>
<td>pure-shear</td>
</tr>
<tr>
<td>PV29</td>
<td>Non prop. shear-comp.</td>
</tr>
<tr>
<td>CS6</td>
<td>Non prop. tension-comp.</td>
</tr>
</tbody>
</table>
Table 6.7: Dimensions and Material Properties of Membrane Elements

<table>
<thead>
<tr>
<th>Speci. Model</th>
<th>Dimensions (mm)</th>
<th>Concrete</th>
<th>Reinforcement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$f'_c$ (MPa)</td>
<td>$\epsilon'_c$ (mm/mm)</td>
</tr>
<tr>
<td>PV19</td>
<td>890x890x70</td>
<td>19.0</td>
<td>0.002</td>
</tr>
<tr>
<td>PV27</td>
<td>890x890x70</td>
<td>20.5</td>
<td>0.002</td>
</tr>
<tr>
<td>PV29</td>
<td>890x890x70</td>
<td>21.7</td>
<td>0.002</td>
</tr>
<tr>
<td>CS6</td>
<td>630x630x100</td>
<td>20.96</td>
<td>0.002</td>
</tr>
</tbody>
</table>

Note: Concrete Initial Stiffness $\frac{E}{\epsilon_c}$

Table 6.8: Results of Tests on Membrane Elements

<table>
<thead>
<tr>
<th>Specimen</th>
<th>Finite Element Analysis</th>
<th>Experiment</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ultimate Load (MPa)</td>
<td>Failure Mode</td>
<td>Ultimate Load (MPa)</td>
</tr>
<tr>
<td>PV19</td>
<td>4.04 $^1$ (3.48) $^2$</td>
<td>Sliding</td>
<td>3.95</td>
</tr>
<tr>
<td>PV27</td>
<td>7.62 $^1$ (6.19) $^2$</td>
<td>Crushing</td>
<td>6.25</td>
</tr>
<tr>
<td>PV29</td>
<td>7.01 $^1$ (7.14) $^2$</td>
<td>Crushing</td>
<td>5.57</td>
</tr>
<tr>
<td>CS6</td>
<td>19.93 $^1$ (19.2) $^2$</td>
<td>Crushing</td>
<td>18.89</td>
</tr>
</tbody>
</table>

1: FEM1 Stress-Based Compressive Softening
2: FEM2 Strain-Based Compressive Softening
Figure 6.28: Load vs. Shear Strain Response, Specimen PV19

Figure 6.29: Load vs. Longitudinal Strain Response, Specimen PV19

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6.30: Load vs. Transverse Strain Response, Specimen PV19

Figure 6.31: Tensile Stress-Strain Response, Specimen PV19
Figure 6.32: Compressive Stress-Strain Response, Specimen PV19

Figure 6.33: Principal Stress Direction vs. Axial Load, Specimen PV19
interesting to see that the analytical model FEM2, which has a strain-based post-cracking compression softening model, predicts the experimental response characteristics better than the stress based model FEM1. The strain components in the longitudinal and transverse directions are shown in figure 6.36 and figure 6.37 respectively. Both the analytical response simulations (FEM1 and FEM2) are almost identical and are more flexible than the experimental response. The tensile stress-strain response shown in figure 6.38 shows a much more rapid reduction of stresses in each of the analyses when compared with the experimental response. The behavior of the analytical models in the compressive stress-strain direction are quite different (figure 6.39). The analytical results FEM2 compare very well with those of the experiment, while the analysis FEM1 is very stiff.

A careful examination of the strength based failure criterion suggests that, with a rapid reduction of tensile stresses, larger compressive stresses are required to cause failure (crushing) as is seen in analysis FEM1. The strength reduction based on strains, is insensitive to the drop in tensile stresses observed in the cracked direction.
(figure 6.38). This suggests that some bond slip may be present, which also explains the occurrence of rather high tensile stresses observed in the experimental response even at high strain levels.

**Panel PV29: Nonproportional loading**

PV29 is a an anisotropically reinforced panel element subjected to a nonproportional loading path. The panel was initially subjected to a pure shear load and then at a load of 3.9MPa a biaxial compressive load was introduced gradually along with the shear load. The tensile stress-strain paths shown in figure 6.40 for both the analyses FEM1 and FEM2 compare very well with the experimental response. It is seen that the tensile stresses are nearly exhausted and the nonproportional load applied at this late stage does not changing the behavior. However the strains in the transverse steel direction continue to grow very rapidly in the experiment, as seen from figure 6.41, even beyond yielding of steel. In the analysis the application of the compressive loads was found to arrest the strain increase and no yielding was registered. The strains in the longitudinal steel direction (which was the stronger direction) showed much better correlation with the experiment (figure 6.42). Both the compressive stress-strain response (figure 6.43) and the shear stress-strain response (figure 6.44) indicate a stiffening after the application of the compressive loads. The reduction of tensile stresses causes the compressive stresses and the ultimate load to be much larger in the analysis FEM1 (ultimate load of 7.1MPa) when compared to the experimental results (5.8MPa). The analytical response FEM2 also appears to have a stiffer response and larger ultimate load(6.9MPa). The inherent path independency of the analytical model could be a reason for some differences. One possible explanation for the observed experimental response behavior is an improvement in the bond transfer capacity due to the application of the lateral compressive loads. The analytical studies are calibrated based on uniaxial test data without any lateral compressive loads present. An increase in bond transfer would possibly cause the transverse steel direction to yield as observed in the experiment, thus making the response more flexible.

**Panel CS6: Non-proportional Loading**

RC panel CS6 is a bi-axially reinforced panel, subjected to non-proportional loads.
Figure 6.35: Load vs. Shear Strain Response, Specimen PV27

Figure 6.36: Load vs. Longitudinal Strain Response, Specimen PV27
Figure 6.37: Load vs. Transverse Strain Response, Specimen PV27

Figure 6.38: Tensile Stress-Strain Response, Specimen PV27
The specimen is initially subjected to cyclic tension along the transverse steel direction, and then subjected to compression in the longitudinal direction keeping the tension in the transverse direction constant. The details of the geometry, material properties, and loading of this panel, tested by Dyngland (1989), is given in tables 6.6 and 6.7. The tensile and compressive stress-strain response of this test specimen is shown in figure 6.45 and 6.46 respectively. While in the experiment the specimen was subjected to cycles of tensile loading, the analysis has been restricted to monotonically increasing loads up to the experimental value (stage 1). From figure 6.45, it is observed that the experimental tensile stress-strain response, excluding the unloading/reloading behavior, is similar to the analytically obtained results. On the application of compression in the longitudinal direction the strains continue to increase in the transverse direction (stage 2). This behavior clearly indicates that some 'Poisson's effect' is present even in cracked reinforced concrete and must be accounted for in analytical models. In the present study, this response has been obtained through the use of post-cracking coupling formulation described.
Figure 6.40: Load vs. Shear Strain Response, Specimen PV29

Figure 6.41: Load vs. Longitudinal Strain Response, Specimen PV29
Figure 6.42: Load vs. Transverse Strain Response, Specimen PV29

Figure 6.43: Tensile Stress-Strain Response, Specimen PV29
in chapter four. The analytical compressive stress-strain response prediction (FEM1 and FEM2) shown in figure 6.46 also compares well with the experimental results. The ultimate loads are shown in table 6.8. The nonproportional loading path in this test specimen appears to be well represented by the analytical model.

6.2.1.3 Summary

A number of panel specimen subjected to different membrane loadings have been analyzed to highlight the influence of the different post-cracking components of the analytical model, viz. cracking, tension-stiffening and its coupling effects, aggregate interlock and softening of the compressive strut. Based on the response predictions of these different specimen under uniform membrane loadings, the following observations can be made:

- The tension-stiffening model including its coupling effects is able to capture the response of most dominant characteristics reasonably well. The tensile
Figure 6.45: Tensile Stress-Strain Response, Specimen CS6

Figure 6.46: Compressive Stress-Strain Response, Specimen CS6
stress-strain responses of specimen in the Dyngland (1989) study (S1, S3, S5, S6, S7), Bhide-Collins (1987) study (PB18, PB19, PB20, PB21, PB22) and the Vecchio Collins (1982) (PV19, PV27, PV29) study are indicative of its capabilities. The parameters used to describe the shape of the tension-stiffening spring are reasonable and have been used for the remainder of the present study. The tensile stress-strain response prediction for specimen CS6, tested by Dyngland (1989), clearly illustrates capabilities of the model in capturing the coupling effects after cracking.

- The aggregate interlock model has been able to capture the sliding component of the deformational response adequately. The shear stress-shear strain response, the rotation of the principal stress direction and the transverse and longitudinal strain responses have all been well represented by the analytical model.

- The softening of the compressive strut has been examined using two different formulations. While the strain based softening formulation performs well in many of the analyzed specimens it has some inherent drawbacks. It suppresses the influence of bond-slip, e.g. specimen PV27, and predicts the compressive stress-strain response well, while the tensile stress-strain response prediction of the specimen is quite different from the experimentally obtained results. Strain based strength and stiffness reduction has also resulted in predicting premature crushing for some specimen, e.g. PV19. The stress-based softening of the cracked concrete compressive strut has resulted in stiff stress-strain responses in some problems. However, this formulation highlights the critical aspects of material response behavior reliably and has been employed in the analytical model for the remainder of this study.

6.2.2 Membrane Loading with Stress Gradients: Shearwalls

Reinforced concrete shearwalls are commonly used in high-rise buildings to provide resistance to wind and earthquake loads. Walls are categorized into two
classes in the ACI code: short walls (Height to width ratio of one), that are primarily designed to mobilize sufficient shear strength to resist loads, and tall walls (Height to width ratios greater then 1.5) that are designed to sustain loads through their flexural capacities. The ACI code provisions are based on uniaxial strength properties of concrete. The amount of reinforcement required in the horizontal and vertical directions are based on a failure mode resulting in the tearing of the tension zone near the base of the wall.

Recent experimental studies of shearwalls by Maier and Thurlimann (1985), and Leifas et al. (1990a) indicate that failure of these walls was due to significant spalling of concrete near the compression toe of the wall. To investigate the influence of the various parameters on the strength of shearwall elements, Leifas et al. (1990a) varied a number of parameters between tests such as the uniaxial strength of concrete, percentages of vertical and horizontal reinforcement, and the amount of dead load present (Vertical loads were applied initially and kept constant) at the time of applying shear loads. From these tests, they concluded that the uniaxial strength properties of concrete was not sufficient to compute the flexural capacity of walls.

As expected the presence of vertical loads in the wall elements stiffened the horizontal load-lateral displacement response of the wall and also increased the ultimate load capacity of the walls. However, it did not seem to alter the mode of failure in these wall elements which was one of failure in the compressive toe region.

Analytical investigations of wall elements, by Wang et al. (1989) indicate that the failure of these elements was due to crushing in the compression toe. In order to obtain the ultimate load of the experiments, these analyses were conducted using modified material properties. The uniaxial strength of concrete was increased and only the material in the portions of the wall near the compression region was permitted to become nonlinear. The rest of the wall was treated as an elastic-cracking solid. This investigation concluded that some parameter other than cracking, post-cracking nonlinearities, and the yielding of the reinforcement was responsible for the observed increase in strength. Analytical studies by Leifas et al. (1990b), indicates that in order to predict the increased strength observed in these walls,
additional reinforcement representing confinement of the concrete, was included in the edge elements. The ultimate load was underestimated without this artificial increase in the reinforcement. They concluded that the height-to-width ratio and the amount of vertical reinforcement was not properly accounted for in the current code recommended provisions for evaluating the shear strength of structural walls.

In the present study, four walls were selected, two from each experimental investigation. The material properties, dimensions of these specimen, and finite element mesh details are presented in tables 6.9 and 6.10 and figures 6.47 and 6.48. Instead of arbitrarily altering the strength or the geometric properties to obtain the experimentally observed ultimate load, emphasis was placed on isolating the dominant phenomenon leading to failure. The load-deflection response obtained in the analyses for each of the specimen are shown in figures 6.49 to 6.52. The experimental and analytical ultimate loads for these specimen are compared in table 6.11. The results indicate that the ultimate strength of the walls obtained in the analysis underestimate the experimental values by as much as 30%. The unconverged point indicated in the analysis (figure 6.49 to 6.52) is due to excessive crushing in the compression toe region which resulted in the inability of the analysis to continue the solution procedure. The deflected shape of the specimen is shown in figure 6.53. The mode of failure obtained in the analysis is by crushing in the compression toe. The compressive stress distribution within the specimen near the ultimate load is shown in figure 6.54. The compressive stresses near the toe were at about the biaxial compressive strength levels (1.16 $f'_c$). The cracking patterns near the ultimate load are shown in figure 6.55. The width of the flexural compressive zone reduces very rapidly near the ultimate. The horizontal and vertical steel stress distributions are shown in figures 6.56 and 6.57, respectively. The vertical steel had yielded in both tension and compression, while the horizontal steel stresses was quite small.

One of these wall specimen, SW21, was selected for a parametric study. The parameters investigated in this study were the different compression softening models in cracked concrete, the threshold angle for multiple cracking (changed from 30° to 90°), and the boundary condition at the base of the beam (changed from a fixed
Figure 6.47: Dimensions of Shearwall Test Specimen
Figure 6.48: Finite Element Mesh For Shearwall Specimen
base to a simple bolting of the beam to the base at four points, thus permitting movement at all other points). The results of these analyses are also shown in figure 6.50. It is seen that none of these parameters appear to have influenced the load deformation-response or the ultimate mode of failure.

Table 6.9: Dimensions of Shearwall Test Specimens

<table>
<thead>
<tr>
<th>Specimen</th>
<th>Dimensions (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
</tr>
<tr>
<td>SW11</td>
<td>300</td>
</tr>
<tr>
<td>Sw21</td>
<td>300</td>
</tr>
<tr>
<td>Maier#4</td>
<td>250</td>
</tr>
<tr>
<td>Maier#9</td>
<td>250</td>
</tr>
</tbody>
</table>

Table 6.10: Material Properties of Shearwall Specimens

<table>
<thead>
<tr>
<th>Speci.</th>
<th>Concrete</th>
<th>Reinforcement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f'_c$ (MPa)</td>
<td>$e'_c$ (mm/mm)</td>
</tr>
<tr>
<td>SW11</td>
<td>44.5</td>
<td>0.002</td>
</tr>
<tr>
<td>Sw21</td>
<td>36.38</td>
<td>0.002</td>
</tr>
<tr>
<td>Maier#4</td>
<td>32.9</td>
<td>0.0023</td>
</tr>
<tr>
<td>Maier#9</td>
<td>29.2</td>
<td>0.0023</td>
</tr>
</tbody>
</table>

The experimentally obtained strength of these wall specimen appear to have been significantly enhanced possibly due to the development of triaxial compressive stresses in the compressive toe region. One possible explanation for the development of these stresses is the influence of the monolithic beam at the base of the wall resisting the development of lateral strains in the wall near the compression toe region. The triaxial stresses developed due to this could cause the increased strength and failure due to crushing and spalling of the concrete in the direction of the least compressive stresses. Another possibility is the presence of confining reinforcement in the beam and the wall edges which could contribute to the triaxial stresses. Even
Table 6.11: Results of Tests on Shearwalls

<table>
<thead>
<tr>
<th>Spec.</th>
<th>Experiment</th>
<th>FE Analysis</th>
<th>ACI-318-89</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Load (KN)</td>
<td>Load (KN)</td>
<td>Flexure Strength</td>
</tr>
<tr>
<td></td>
<td>Failure Mode</td>
<td>Load (KN)</td>
<td>$f_{wu}(\text{KN})$</td>
</tr>
<tr>
<td>SW11</td>
<td>260</td>
<td>crushing</td>
<td>182</td>
</tr>
<tr>
<td>SW21</td>
<td>127</td>
<td>crushing</td>
<td>83</td>
</tr>
<tr>
<td>Maier#4</td>
<td>370</td>
<td>crushing</td>
<td>305</td>
</tr>
<tr>
<td>Maier#9</td>
<td>325</td>
<td>crushing</td>
<td>268</td>
</tr>
</tbody>
</table>

1: A vertical load of 260KN is applied initially and kept constant.

Figure 6.49: Load Displacement Response of Shearwall Specimen SW11
Figure 6.50: Load Displacement Response of Shearwall Specimen SW21

Figure 6.51: Load Displacement Response of Shearwall Specimen Maier#4

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
a small compressive stress of $5\% f'_c$ in the least compressive stress direction could increase the strength (triaxial compressive strength) of the specimen by as much as 20% due to the triaxial state of stress. It has been speculated (Kotsovos, 1988) that the dilatation present near compressive failure at some points would be resisted by the neighbouring points that are cracked, enhancing the strength of the wall element. Another factor that may be partially responsible for the underestimation of the ultimate load of the walls is the strain hardening in steel. However, complete data on the strain hardening properties of the reinforcement were not available and therefore not included in the analysis.

The ultimate strength predictions of these walls, in flexure and shear, using the ACI code are also shown in table 6.11. These results indicate that the estimated flexural capacity of these wall specimen, based on the ACI code, is exceeded in these tests. The contribution of the vertical reinforcement in sustaining the flexural loads has not been well accounted for in the ACI code. Further, truss analogy does not appear to be well suited for the analysis and design of shearwalls. The shear
Figure 6.53: Deflection Profile, Specimen SW21
Figure 6.54: Compressive Stress Distribution in Shearwall Specimen SW21 Near Ultimate Load
Figure 6.55: Crack Pattern in Shearwall Specimen SW21 Near Ultimate Load
Figure 6.56: Stress Distribution in Horizontal Steel Near Ultimate Load, in Shear-wall Specimen SW21
Figure 6.57: Stress Distribution in Vertical Steel Near Ultimate Load, in Shearwall Specimen SW21
capacity of these walls, predicted by the ACI code, considerably overestimate the strength of these wall specimen.

6.3 RC Elements Subjected to Out-of-Plane Loadings

Delft Beam
This moderately deep beam, tested experimentally by Walraven (1978), belonged to a group of beams tested to study the influence of beam depth on the shear strength of concrete. The beam was simply supported at the ends and subjected to concentrated loads at points 500mm on either side of the center. In the experiment it was observed that vertical cracks initially formed in the pure moment zone of the beam. As the loads were increased, new inclined cracks formed between the support and the loading point. It was reported that the final failure of the beam was due to the penetration of these inclined cracks deep into the compression zone resulting in a sudden failure. The material properties have been given in table 6.12. The beam geometry details are shown in figure 6.58 and the finite element mesh, along with the layering details are shown in figure 6.59.

In the present study this beam was analyzed using both the implicit layering procedure and the explicit layering procedure discussed in section 5.5. For the implicit layering analysis eleven layers, ten concrete layers and one steel layer, were used to discretize the cross-section. The concrete layers adjacent to the steel layers were tension-stiffening layers (each 15mm thick) while the other layers were strain-softening layers. The fracture energy \( G_f \) used in each strain-softening layer was 60N/m (deBorst and Nauta, 1985). The number of layers selected to discretize the cross-section is based on a previous study by Abdel-Rehman (1981), which concludes that between six to ten layers was sufficient to capture the response behavior reasonably. For the first analysis a relatively coarse mesh of seven elements along the length was selected to model half of the beam, taking advantage of symmetry (figure 6.59), using the implicit layering procedure. A 3x3x1 Gaussian numerical integration procedure has been employed to compute the stiffness of both the steel and concrete layers. The load vs. deformational response (center point deflection) is
shown in figure 6.60 along with the experimental results, and results from an analytical study by deBorst and Nauta (1985). It was found that at a load of nearly 70 KN the beam 'failed' due to lack of convergence. The analytical response obtained by deBorst and Nauta (1985) indicated a much higher load carrying capacity (nearly 80 KN). They observed that in obtaining these results it was necessary to restrict the threshold angle required for the formation of multiple cracks at a point to 60°. The use of lower threshold angles (30°) resulted in a lack of convergence of the nonlinear procedure beyond a load of 70 KN. In the analysis with a refined mesh, deborst and Nauta (1985) observed a small reduction in the stiffness at a load of 70 KN but the ultimate failure was due to yielding of the reinforcement leading to a collapse of the tied arch mechanism. In the present study, a refined mesh (12 elements) was employed and the analysis was repeated. The results of this are also shown in figure 6.60. In this analysis, at approximately 70 KN, a kink in the load-deflection response was observed. On increased loading, this instability point was passed and the beam was able to sustain additional loads, failing eventually at 77 KN due to yielding of the steel. The deflected shape of the specimen is shown in figure 6.63. The cracking pattern for this analysis is shown in figure 6.66. The initial vertical cracks were in the pure moment zone, but when the loading was continued, new inclined cracks developed between the support and the loading point. These cracks penetrated deep into the compression zone near the ultimate load. The compressive stress distribution at the top surface of the concrete is shown in figure 6.64. The stress distribution in the reinforcement near ultimate load is shown in figure 6.65. The concrete in-plane normal and transverse shear stress distribution through the cross-section of the beam are shown in figures 6.61 and 6.62, respectively, at a point 1340mm from the support of the beam. As expected prior to cracking, the concrete normal stress distribution is linear while the transverse shear stress distribution is parabolic. At a load of 77KN (ultimate load), very small tensile normal stresses are present near the steel layer, and vanish completely in the lower two thirds of the beam cross-section due to cracking. Compressive stresses are mobilized in the top one third of the beam. It should be pointed out that the shear stresses plotted do not go up to the surface of the beam where they are expected to vanish.
Table 6.12: Material Properties of RC Elements Subjected to Out-of-Plane Loadings

<table>
<thead>
<tr>
<th>Specimen</th>
<th>Concrete</th>
<th>Reinforcement Mesh¹</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f'_c$ (MPa)</td>
<td>$\varepsilon'_c$ (mm/mm)</td>
</tr>
<tr>
<td>ML9</td>
<td>44.4</td>
<td>0.0025</td>
</tr>
<tr>
<td>Delft Beam²</td>
<td>35</td>
<td>0.002</td>
</tr>
<tr>
<td>Dudeck Slab</td>
<td>43.01</td>
<td>0.0027</td>
</tr>
<tr>
<td>Regan Slab1#2</td>
<td>24.8</td>
<td>0.002</td>
</tr>
</tbody>
</table>

1: Two way orthogonal reinforcement
2: Bottom steel along Beam length only

Figure 6.58: Dimensions of Specimen Delft Beam

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6.59: F.E. Mesh and Layering details for Specimen Delft Beam
Figure 6.60: Load Vs. Deflection Response, Specimen Delft Beam
Figure 6.61: Normal Stress Distribution at Point, \( x=1340 \text{mm} \), from the Support, Implicit Layering Procedure

Figure 6.62: Transverse Shear Stress Distribution at a Point, \( x=1340 \text{mm} \), from the Support, Implicit Layering Procedure
Figure 6.63: Deflection Profile, Specimen Delft Beam, Implicit Layering.
Figure 6.64: Concrete Compressive Stresses-Top Surface, Specimen Delft Beam, Implicit Layering.
Figure 6.65: Reinforcement Stresses, Specimen Delft Beam, Implicit Layering.
Figure 6.66: Cracking Pattern, Specimen Delft Beam, Implicit Layering.
The deflected shape obtained in the analysis using explicit layering is shown in figure 6.67. The compressive stress distribution on the top surface of the beam is shown in figure 6.68. It is interesting to see that the stress distribution in the flexural zone is not uniform leading to localized crushing under the loads. The stress distribution in the reinforcement is shown in figure 6.69. The crack patterns from the explicitly layered analysis is shown in figure 6.70. These cracks initially develop in the pure moment zone but on subsequent loading some inclined cracks develop in the region between the support and the load. Some slightly inclined cracks appear at a few points in the pure moment zone. A possible explanation for this could be the relatively coarse mesh employed between the loading point and the center (two elements along the length). Thus some boundary effects could cause these shear stresses, especially after cracking. The point directly under the load indicates excessive cracking due to very high localized shear stresses. The load-deflection response from this analysis is shown in figure 6.60. Two different analysis were conducted using different amounts of fracture energy. The smaller amount of 60 N/m resulted in premature crushing. But on increasing the fracture energy to 400 N/m a slightly stiffer response was observed leading to localized crushing failure at nearly 80 KN. This clearly indicates that the explicit layering procedure is much more sensitive to the employed material parameters employed then the implicit layering procedure. In the present study, only implicit layering procedures was used for analyzing flexural test specimen.

**Torsional Element ML9**

In general reinforced concrete slab elements are subjected to flexural and torsional loads in the reinforcing directions. In lightly reinforced flexural elements a yield line analysis approach is able to predict the ultimate load with reasonable accuracy. In structural elements with large amounts of reinforcement (1.5% or more) such an analysis procedure results in unconservative estimates of the ultimate capacity. The reason for this is that crushing in concrete precedes the yielding in the reinforcement. As discussed in the preceding sections, diagonally compressed concrete elements undergo considerable softening due to cracking (Vecchio and Collins 1982, 1986). In structural elements that are heavily reinforced, even the lower bound analytical
Figure 6.67: Deflection Profile, Specimen Delft Beam, Explicit Layering.
Figure 6.68: Concrete Compressive Stresses-Top Surface, Specimen Delft Beam, Explicit Layering.
Figure 6.69: Reinforcement Stresses, Specimen Delft Beam, Explicit Layering.
Figure 6.70: Cracking Pattern, Specimen Delft Beam, Explicit Layering.
approaches are questionable as they do not account for these effects. Marti et al. 1987a, 1987b, tested a number of heavily reinforced slab elements (nearly 2%) under torsional loads. In this study one specimen from this series (ML9) was selected and analysed.

The material parameters of the specimen is shown in table 6.12. The slab is a square of sides 1600mm and 200mm thick, having a reinforced mesh on the top and bottom with a cover of 20mm. The torsional moments generated in the experiment by applying equal and opposite loads at adjacent corners were simulated analytically by applying uniform edge moments along the line joining the mid points of the slab sides. Accordingly, the isotropic reinforcement mesh was rotated 45° (from its original 0° - 90° orientation). A single element was used to describe the specimen since the stress field is uniform. The implicit layering procedure was employed to model the cross-section with concrete and steel layers at appropriate positions. A total of fourteen layers were used to discretize the section, of which four layers represent the steel with appropriate orientations (45° and 135° to the loading), while the remaining layers were used to describe the concrete. A 3x3x1 Gaussian numerical integration procedure was used to compute the stiffnesses of both concrete and steel layers. After cracking, tension stiffening was considered in only concrete layers adjacent to the steel meshes (two orthogonally placed adjacent steel layers) and were four layers in all. The thickness of the tension-stiffening layers is 20mm each. The remaining concrete layers were assumed to be strain-softening layers after cracking. Thus each concrete layer was approximately 20mm thick. A threshold angle of 15° for the initiation of new cracks, with respect to previously existing cracks, was employed in the analysis.

The analytical prediction of the moment vs. the principal curvature for this specimen is shown in figure 6.71 along with the experimental results. The analytical prediction is reasonably close to the experimental results. The sudden increase in the curvature seen in the analytical response at a moment of 40 KN-m/m is due to cracks propagating through the thickness of the slab element rapidly. The experimental results were not available at such small load increments. The mode of failure of this specimen was due to crushing of concrete on both slab faces and
not due to yielding of steel, which is also observed in the analysis. The analysis predicted cracks penetrating to the central two layers (7 and 8) from each face of the slab that are mutually orthogonal. Thus, layers 7 and 8 cracked orthogonally and are in a state of biaxial tension. The neutral axis in each principal direction shifted to the level of the fifth layer from the compression face from its initial position at the center of the section. The load carrying capacity of the slab after cracking was mobilized in the outer four layers (1-4 and 11-14) in each principal direction. The ultimate moment obtained in the present study was 95 KN m/m applied along the edge of the slab versus an experimental value of 104.8 KN m/m. Marti et al. 1987a and 1987b, have also reported the ultimate values obtained from yield line analysis (136 KN m/m), lower bound (114 KN m/m), Compression field theory (Collins and Mitchell 1980, 92KN m/m) and ACI (46.4 KN m/m). It appears that the ACI code predicts an extremely conservative estimate of the ultimate load, while estimates based on either lower bound or yield line analysis are unconservative. The results based on the compression field theory (Collins and Mitchell 1980) and those
obtained in the present study account for the compressive strength and stiffness reduction after cracking and are able to predict reasonable values of the ultimate load.

**Dudeck Slab S1**

RC Slab S1 was an isotopically reinforced slab tested by Dudeck et al. (1978). This slab was supported only at the corners where the transverse deflection was restrained and loaded at the center by means of a concentrated load. Taking advantage of symmetry, one quarter of the slab was analyzed in this study. The dimensions, finite element mesh discretization and the details of the implicit cross-sectional layering procedure is presented in figure 6.72 and 6.73 respectively. The stiffness of concrete and steel layer was computed using a 3x3x1 Gaussian integration scheme. The material properties used in analyzing this specimen are given in table 6.12. A threshold angle of 30°, between any two cracks at the same point was employed for the analysis.

The load-deflection response obtained in the present study is shown in figure 6.74 along with the experimentally obtained results. Figure 6.74 also shows the analytical results obtained by Abdel-Rehman (1981) and Milford and Schnobrich (1984). Abdel Rehman (1981) employed a fixed crack model with a 30° threshold angle between cracks, while Milford and Schnobrich (1984) employed a rotating crack model in their study. The load-deflection response obtained in the present analysis is close to the experimental results up to yielding in the reinforcement. The analysis detected some localized crushing, resulting large number of iterations for convergence. Thus, the analysis was terminated at this stage (53KN- 'ultimate load'). The ductility observed in the experimental results and other analytical investigations may also be due to strain hardening in the reinforcement. The deflected shape of the specimen is shown in figure 6.75. The crack patterns obtained in the present study are shown in figure 6.76 and the yielding of the reinforcement in the bottom layers is shown in figure 6.77 and 6.78. The mode of failure obtained in the analysis was by yielding of the tension reinforcement. The ultimate load obtained in the present study is nearly 53KN, while the experimental value is 61.6KN. The failure loads obtained by Abdel Rehman (1981) and Milford and Schnobrich (1984) are
58KN and 60KN, respectively. The ultimate load obtained using yield line analysis is 56KN.

Figure 6.72: Dimensions of Specimen Dudeck Slab S1

**Regan Slab series 1#2**

This reinforced concrete slab, tested by Regan (1986), was simply supported at the four edges, having a downward load applied at the center by means of a monolithic stud having dimensions 200mmx200mm. In this study the load was applied directly on the slab. The details of the geometry, the implicit layering details and finite
Figure 6.73: F.E. Mesh and Layering Details for Specimen Dudeck Slab S1
element mesh are shown in figures 6.79 and 6.80. A 3x3x1 Gaussian numerical integration procedure was employed to compute the stiffness contributions of each steel and concrete layer. The material properties are given in table 6.12. The tension-stiffening layer, on each side of the reinforcement, had a thickness of 8mm. A threshold angle of 30° degrees was used in this analysis for the initiation of additional cracks with respect to previously existing cracks at the point. This slab belonged to a series of slabs tested by Regan (1986) to determine the influence of various parameters on 'punching' failures in slabs near the column region. The reinforcement in this particular specimen was uniformly distributed across the slab in the tension zone. The load displacement response for this specimen obtained from the analysis is shown in figure 6.81 along with the experimental results. The mode of failure obtained in the analysis was concrete crushing in the top surface (compression) followed by yielding in the tension steel. No convergence could be obtained at this stage, as seen in figure 6.81. The deflected shape of the specimen is shown in figure 6.82 along with the undeformed shape.

Figure 6.74: Load Vs. Deflection Response, Specimen Dudeck Slab S1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6.75: Deflection Profile, Specimen Dudeck Slab S1
Figure 6.76: Cracking Pattern, Specimen Dudeck Slab S1, Bottom Surface.
Figure 6.77: Yielding in Bottom Reinforcement, Specimen Dudeck Slab S1, X Direction.
Figure 6.78: Yielding in Bottom Reinforcement, Specimen Dudeck Slab S1, Y Direction.
Figure 6.79: Dimensions of Specimen Regan Slab 1#2
Figure 6.80: F.E. Mesh and Layering Details for Specimen Regan Slab 1#2
The final cracking pattern is shown in figure 6.83 and 6.84. The cracks near the ultimate load are shown over different cross sections of the slab. The loading in these tests was similar to an inverted column slab connection, resulting in large circumferentially and radially cracked region at the slab bottom surface. The size of these ring cracks are reduced near the mid surface of the slab cross section. This clearly indicates the development of the failure 'cone', typical of the punching mechanism. The cracks appear to reach the mid-surface of the slab. The minimum principal stresses on the compression face of the slab near ultimate is shown in figure 6.85. The stress distribution in the reinforcement is shown in figures 6.86 and 6.87. The implicitly layered analytical solution appears to predict both the load-deflection response and the stress distribution adequately for this slab. In the experiments, Regan (1986) found that the variation in the distribution of the reinforcement had no influence on the load carrying capacity of the slabs. However, the deflection in slabs with reinforcement closer to the loaded area (connection region) was observed to be somewhat less. This indicates that the reinforcement in
the 'column strip' of the slab is a better measure of the percentage of reinforcement required in design as compared to using the entire slab width between column centerlines. The ultimate load for this slab obtained in the analytical study was 146 KN (implicit layered analysis). This was approximately 84% of the experimental value of 176 KN. Regan (1986) also reported the predictions obtained from four different codes- British standard 8110, CP110, ACI 318 and CEB-FIP. The critical failure sections considered for this analysis by Regan (1986), as per the different codes, for these prediction is shown in figure 6.88. The failure values obtained from these analyses are 178 KN (BS 8110), 172 KN (CP 110), 138 KN (ACI 318) and 134 KN (CEB-FIP). Yield line analysis of this slab resulted in a failure load of 136 KN as reported by Regan (1986). The values predicted by the British codes (BS 8110 and CP 110) are based on larger failure zones, compared with the ACI and the CEB-FIP codes. Additionally the limiting shear stress at the failure zone is a function of only concrete strength in the ACI code, while the remaining codes include the contribution of the flexural reinforcement and the slab depth. The safety factors built into the respective codes were not considered for the purpose of this comparison. The analytical failure prediction obtained in this study, was about 10% higher than the yield line prediction indicating that the contribution of the compressive strength of concrete, which undergoes some strength reduction due to cracking, is significant. The failure mode appears to be 'punching' rather then yielding of the slab. The ACI and the CEB-FIP predictions are almost equal to that of the yield line analysis prediction. The predictions obtained using the British codes are much closer to the experimental value which could be due to the larger critical failure section considered. The cracking patterns obtained in this study (6.83 and 6.84) indicated that the 'failure surface' is about 135mm from the face of the 'column' (loaded area), which corresponds to a value of 1.35 times the slab thickness (1.35d).
Figure 6.82: Deflection Profile, Specimen Regan Slab 1#2
Figure 6.83: Cracking Pattern, Specimen Regan Slab 1#2, Bottom Surface.
Figure 6.84: Cracking Pattern, Specimen Regan Slab 1#2, Middle Surface.
Figure 6.85: Minimum Principal Stress Distribution on the Compression Face, Specimen Regan Slab 1#2, At the Ultimate Load.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6.86: Stress Distribution in the Reinforcement, X-Direction, Specimen Regan Slab 1#2
Figure 6.87: Stress Distribution in the Reinforcement, Y-Direction, Specimen Regan Slab 1#2
Figure 6.88: Critical Section, Around Columns, Employed in Different Codes.
6.4 RC Slab Element Subjected to General Loading

Slab Elements examined up to this point in this study have been simply supported at the edges and subjected to an axial load in the central region, simulating a column slab connection. Such loadings result in either a ductile load-deflection response and failure due to yielding of the tension reinforcement (eg. Dudeck slab S1, where the reinforcement percentages are low), or a brittle crushing failure at the center of the slab characterized by punching of the 'column' through the slab along with a portion of the slab (eg. Regan slab 1#2 where reinforcement levels are moderately high). While design codes address the need to have design criteria to restrict the possibilities of brittle failures, they do not have provisions to account for the influence of 'lateral restraint' on the punching capacities of slabs, developed due to boundary restraint at the edges. In the experimental study conducted by Regan (1986) a group of slabs that were subjected combinations of uniform edge moments and a central axial load were examined. These slab specimen have a shape that is like a cross in plan, with a 1.73m square central panel and projections of 635mm on each side. The details and other dimensions are provided in figure 6.89.

An upward load was applied at the center of the slab by means of a 160mm square plate, while downward line loads were applied at the sides of a 1.83m square. The assembly is supported by means of rollers 457mm beyond the downward line load. By adjusting the ratios of the downward line load and the upward central load various levels of restraining moment were generated along the sides of the 1.83m square. This moment to load ratio was kept constant through the test for each slab, while it was varied between slabs. In this study one specimen from this series was analyzed with the moment per unit width to load ratio (\(m/F\)) of 0.036. One quarter of this specimen has been analyzed using the implicit layering procedure, taking advantage of symmetry. The details of the finite element mesh are provided in figure 6.90. The material properties of the constituents are given in table 6.13. A threshold angle of 30° for the initiation of new cracks with respect to previously existing cracks was employed for this analysis. The central square portion of the slab having an orthogonal mesh reinforcement at both the top and bottom region of the slab
Table 6.13: Material Property of RC Slab Element Subjected to General Loadings

<table>
<thead>
<tr>
<th>Specimen</th>
<th>Concrete</th>
<th>Reinforcement Mesh</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$f'_c$ (MPa)</td>
<td>$\epsilon'_m$ (MPa)</td>
</tr>
<tr>
<td>Regan Slab 4#3</td>
<td>34.1</td>
<td>0.0023</td>
</tr>
</tbody>
</table>

Figure 6.89: Dimensions of Specimen Regan Slab 4#3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6.90: F.E. Mesh Details for Specimen Regan Slab 4#3

(cover of 23mm on each face) was represented using ten layers, four steel and six concrete layers. The projections of the slab, marked ABCD and DEFG respectively in figure 6.89, was uni-directionally reinforced at both the top and bottom surface (effective cover of 23mm) respectively. These portions of the slab were represented by eight layers through the cross section, two steel and six concrete layers. The tension-stiffening layer, one on each side of the steel mesh, was 8mm thick.

Figure 6.91 shows the load vs. strain response on the compressive surface of the slab, at a point on the diagonal 95mm from the central loading. The average rotations of the top surface (calculated with respect to the deflection at the center of the slab) with increasing loads is shown in figure 6.92. The deflection profile is shown in figure 6.93. The cracking pattern at 30% of the experimental ultimate load is shown in figure 6.95 and 6.96. The analytical results were obtained up to 50% of the experimentally observed ultimate. However, at this stage the number of cracks initiated increased significantly and the analytical procedure was terminated as it became computationally cost prohibitive. The load strain response and
the load rotation response indicate that the analytical model is close to the experimental results. One possible reason for the increased flexibility and excessive cracking observed in the load-strain response in the analysis is a reduced participation of the tension-stiffening response. This may be due to a very thin layer of concrete (8mm) being considered as a tension-stiffening layer on either side of the reinforcement. The crack patterns indicate that due to the upward central load, radial and circumferential cracks are seen in the top surface of the slab near the center. The lower surface has flexural cracks along the line of action of the downward loads. The compressive stresses in the bottom surface of the slab is shown in figure 6.94. Experimental evidence presented by Regan (1986) indicated that even for small ratios of edge moment to punching load, the punching strength of the slab elements increased by as much as 25% capacity when no boundary restraints were placed. The analytical model was able to predict the dominant deformational response characteristics up to half the ultimate load.
Figure 6.91: Load Vs. Strain Response, Specimen Regan Slab 4#3, At Corner of Central Load.

Figure 6.92: Load vs. Rotation Response, Specimen Regan Slab 4#3, At Point Outside Central Load (x=280mm and x=480mm)
Figure 6.93: Deflection Profile, Specimen Regan Slab 4#3
Figure 6.94: Compressive Stress Distribution, Specimen Regan Slab 4#3, Bottom Surface

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6.95: Cracking Pattern, Specimen Regan Slab 4#3, Top Surface.
Figure 6.96: Cracking Pattern, Specimen Regan Slab 4#3, Bottom Surface.
Chapter 7

Summary and Conclusions

7.1 Summary

A three dimensional nonlinear inelastic finite element model for the analysis of reinforced concrete structures has been developed in this study. The treatment of concrete prior to cracking is simulated using a nonlinear elastic secant stiffness formulation proposed by Ottosen (1979). A strength envelope proposed by Ottosen (1977) has been employed in conjunction with this formulation to detect failure in concrete. The stress-strain response of concrete has been ratified in the present study by comparing the analytical results with available test data on concrete response to multiaxial states of stress. Steel has been simulated by employing an elasto-plastic formulation with possible strain-hardening, having uniaxial stiffness properties.

Cracking in Concrete has been simulated using a non-orthogonal multiple cracking formulation proposed by deBorst and Nauta (1985). The plain concrete stress-strain response after cracking has been simulated using a bilinear curve in the crack normal direction, similar to that employed by Rots et al. (1985). To include the tension-stiffening effects in cracked reinforced concrete, a stress strain relationship in each reinforcing direction (considered as part of concrete), has been developed in the present study. For the simulation of aggregate interlock response in reinforced concrete, a variable crack interface shear stress-strain response which simulates the effects of shear-friction has been proposed in this study. The reduction in the stiffness of the cracked concrete compressive strut, observed in tests by Vecchio and Collins (1982), has been considered in this study. Two formulations to simulate these effects have been employed in this study: one based on strains proposed by Vecchio and Collins (1982, 1986) and a second one based on stresses advanced in this study.
To simulate the variations in material properties through the cross-section of an RC component, two cross-sectional layering schemes have been employed. One formulation is based on implicit procedures, similar to that by Figueras and Owen (1983), while the other is based on explicit procedures proposed by Barzegar (1989).

The capabilities of the analytical model developed in this study have been examined by analyzing the response of RC panels, walls, and slab specimen under the action of various loads.

7.2 Conclusions

The tension-stiffening formulation including its coupling effects proposed in this study is able to represent the post-cracking tensile stress-strain response of reinforced concrete test specimen adequately. The panel elements analyzed in the present study indicate that this formulation is invariant with respect to transformations (panels S1, S3, S6, - Dyngland, 1990) and size effects (bars Shima #3, Shima #5, - Shima 1987; panels PB13, - Bhide and Collins 1987, Rizkalla #2, - Rizkalla and Hwang 1984). The representation of tension-stiffening effects in each steel direction, instead of in the crack normal direction, ensures that tensile stresses are transferred through the concrete even after the reinforcement in some direction(s) have yielded (PV19 - Vecchio and Collins 1982). The present formulation is simple and depends only on uniaxial properties of concrete and steel.

The compressive stress-strain response of the cracked concrete strut is well represented by the stress-based strength and stiffness reduction procedure proposed in this study. The strain-based procedure (Vecchio and Collins 1982) tends to overestimate the reduction of strength of the strut (PV19). In addition, the strain-based procedure cannot detect the presence of bond slip (PV27, PV29 - Vecchio and Collins 1982).

The representation of the crack interface shear stress-strain response, using a variable shear stiffness procedure, proposed in this study is adequate for capturing the response of RC test specimen. The suitability of this model is demonstrated by simulating the response of uniaxially reinforced specimen (PB18 to PB22 - Bhide.
and Collins 1987), where the role of the crack interface shear stiffness in capturing the dominant response behavior is significant.

The layering procedures implemented in this study, both implicit and explicit, have been adequate in tracing the nonlinear response behavior through the depth of reinforced concrete specimen. The distribution of stresses through the cross-section is well represented by the implicit layering procedure, even for a moderately thick specimen, and is suitable for inclusion in material model computations. The explicit layering procedure introduces a flexibility through the cross-section due to additional degrees of freedom. This may be important for very thick cross-sections. However, the ultimate loads of the reinforced concrete specimen analyzed in this study do not appear to be altered significantly due to this shear flexibility (Delft Beam - Walraven 1978), indicating that the implicit layering procedure is suitable for nonlinear analysis to predict the ultimate loads. For thin specimen the implicit layering procedure is able to predict the displacements and strains reasonably.

The investigations in the present study indicate that the analysis and design of reinforced concrete shearwalls, which are based on uniaxial properties of concrete and steel, are inadequate in estimating the ultimate loads. It appears that the multiaxial states of stress present in the compression region of the wall need to be considered in the design of shearwalls.

The response of RC flexural and torsional specimen in the analyses clearly highlights the role of the reduced compressive strength of cracked concrete in altering the ultimate load, the load-deformational response and the mode of failure in these specimen (ML9 - Marti et al. 1987; slab S1 - Duddeck et al 1978; Regan Slab 12 and 43 - Regan 1986). Analysis procedures such as the yield line and the lower bound theory do not account for this nonlinearity and are thus unable to estimate the ultimate load adequately.

The influence of lateral restraint on the response of slab elements, in particular their “punching capacity”, is well represented by the analysis in the present study up to a point. The code provisions do not account for the effect of boundary restraints in estimating the ultimate capacity of connection elements.
More investigations are needed to evaluate all the parameters influencing the load transferring capacity of the connection regions.

7.3 Recommendations for Future Work

Based on the present study, some of the areas where some further work needs to be done are identified below:

1. Experimental studies need to be conducted to determine the influence of lateral confinement on the tensile stress-strain response of RC test specimen. This would provide data to simulate 'tension-stiffening' at various levels of lateral confining stresses.

2. Develop models to consider the influence of shear dilatancy on the stress-strain response of reinforcement and tension-stiffening response. The present study considers the influence of the confining pressure due steel and tension-stiffening 'bond' on the crack interface shear response (shear-friction).

3. Develop numerical procedures that enable the use of larger load increment sizes, for use in secant stiffness models. This may involve a procedure to prevent artificial softening because of the iterative approach.
Bibliography


200


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


Vecchio, F.J. and Collins, M.P.(1982). "The Response of Reinforced Concrete Elements to In-plane Shear and Normal Stresses". Department of Civil Engineering, University of Toronto, Publication 82-03.


Walraven (1978). "The Influence of Depth on The Shear Strength of Light Weight Concrete Beams Without Shear Reinforcement, Report 5-78-4, Stevens Laboratory, Delft University, Netherlands.


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
NONLINEAR INELASTIC FINITE ELEMENT ANALYSIS OF REINFORCED CONCRETE STRUCTURES WITH EMPHASIS ON SHEAR AND TORSION

Volume II

A Dissertation

Submitted to the Graduate Faculty of the Louisiana State University and Agricultural and Mechanical College in partial fulfillment of the requirements for the degree of Doctor of Philosophy

in

The Department of Civil Engineering

by

Ananth Ramaswamy
B.Tech., Indian Institute of Technology, 1985
M.S. University of California, 1986
May, 1992

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Appendix A
Flow Chart

START

RESTART

PRE-PROCESS INPUT DATA

INCREMENT LOOP

ITERATION LOOP

SETUP LOAD & STIFFNESS VECTOR

SOLVE FOR DISPLACEMENTS

RECOVER STRESS AND STRAIN

UPDATE STIFFNESS

ITERATIONS EXCEEDED?

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

YES

STOP

YES

STOP

NO

NO

PRE-PROCESS INPUT DATA

INCREMENT LOOP

ITERATION LOOP

SETUP LOAD & STIFFNESS VECTOR

SOLVE FOR DISPLACEMENTS

RECOVER STRESS AND STRAIN

UPDATE STIFFNESS

ITERATIONS EXCEEDED?

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?

NO

STOP

YES

STOP

NO

CONVERGENCE CHECK?

NO

MORE INCREMENTS?
Appendix B
Sample Input Data File

OFF='CERAMA.DE5ELA' VOLSER='USER07' KEEP ! NAMES OF OUTPUT FILES
SCFP = 'CERAMA.DE5ELA' ! NAMES OF SCRATCH FILES STATUS KEEP
BEGIN ! SETUP INPUT FILES FOR PROCESSING
DIMENSION 3 ! 3 D PROBLEM
LINEAR ! GEOMETRIC LINEARITY
SYMMETRIC ! SYMMETRIC SOLVER
INCREMENTS 40 FRACT 0.4000 ITERATIONS 80 ICKNTR 20 ! INCREMENTS, LOAD FRACTION
INCREMENTS 35 FRACT 0.5000 ITERATIONS 80 ICKNTR 20 ! NUMBER OF ITERATIONS
INCREMENTS 35 FRACT 0.6000 ITERATIONS 80 ICKNTR 20 ! CRACK OUTPUT EVERY 20
INCREMENTS 30 FRACT 0.6000 ITERATIONS 80 ICKNTR 20 ! STEPS.
INCREMENTS 40 FRACT 0.7000 ITERATIONS 80 ICKNTR 20
INCREMENTS 40 FRACT 0.8000 ITERATIONS 80 ICKNTR 20
INCREMENTS 35 FRACT 0.9000 ITERATIONS 80 ICKNTR 20
INCREMENTS 30 FRACT 1.0000 ITERATIONS 80 ICKNTR 20
MATERIAL 1 TYPE 4 ! MATERIAL TYPE 4 REINFORCED CONCRETE
ELEMENT 1 TO 7 TYPE 3408 MATERIAL 1 THICK 6 ! ELEMENT TYPE AND PROPERTY
CAEDS 'CERAMA.DE5ELA'(UNV) ! CAEDS UNIVERSAL FILE PROCESSED FOR NODES AND ELEMENTS
LOADS ! LOADING TYPE AND INFORMATION
NODE 216 FZ -1.333333
NODE 217 FZ -5.333333
NODE 218 FZ -1.333333
END
OUTPUT 1 DISPLACEMENTS REACTIONS CONCRETE ! OUTPUT INFO FOR REINFORCED CONCRETE
ELEM 1 TO 42 DONE NODE 1 TO 228 DONE END ! AT SPECIFIC ELEMENTS AND NODES
RCPREP ! CALL TO MATERIAL MODEL INPUT PROCESSOR
SECANT ! SECANT MODEL
MATERIAL ! MATERIAL 1 TYPE - E.G. CONCRETE AND INFORMATION
CONCRETE
E 28000.0 NU 0.20 FC 28.0 FT 2.50 EPSG 0.002 EPSU 0.0030 DSOFT 0.0001
BSER 0.99 OF 0.40 BUF 0.46 BETA 0.8 CFSIG 0.85 ATETA 80. TSTFLAG -99.0
TSTERM 0.003 TOLER 10. TSTIFG 1 CSDFTN 2
MATERIAL ! MATERIAL 2 TYPE - STEEL AND INFORMATION
STEEL
E 210000.0 NU 0.0 YEILD 440.0 AYER 0.001 EPSBRK 0.022
ELEM 1 TO 7 ! LAYERING INFORMATION-NUMBER OF INTERNAL LAYERS AND DESCRIPTION
LMAT 1 LTHICK 13.45 ZI -2.375 HIPXI 3 HIPETA 3 HIPSI 2
ORIENT 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 LOCATION ZI,NO. OF GAUSS PTS.
LMAT 1 LTHICK 15.00 ZI -9.06 NIPXI 3 NIPETA 3 NIPSI 2 ! ORIENTATION OF LAYER ETC.
ORIENT 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
LMAT 2 LTHICK 3.1 ZI 0.0 HIPXI 3 HIPETA 3 HIPSI 2
ORIENT 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
LMAT 2 LTHICK 3.1 ZI 0.0 NIPXI 3 NIPETA 3 NIPSI 2
ORIENT 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
LMAT 2 LTHICK 15.00 ZI 9.08 HIPXI 3 HIPETA 3 HIPSI 2
ORIENT 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
LMAT 2 LTHICK 15.00 ZI 3.275 HIPXI 3 HIPETA 3 HIPSI 2
ORIENT 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
DONE
ELEM 8 TO 42 ! REMAINING ELEMENTS HAVE ONE LAYER OF TYPE 1 - CONCRETE
LMAT 1 LTHICK 78.00 ZI 0.0 HIPXI 3 HIPETA 3 HIPSI 2
ORIENT 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
DONE
END
CONSTRAINT ! MULTIPLE CONSTRAINTING DONE EXTERNALLY.

212
<table>
<thead>
<tr>
<th>Stack Element</th>
<th>From</th>
<th>To</th>
<th>By</th>
<th>Stop</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>STOP</td>
</tr>
<tr>
<td>37</td>
<td>2</td>
<td>7</td>
<td>1</td>
<td>STOP</td>
</tr>
<tr>
<td>38</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>STOP</td>
</tr>
<tr>
<td>39</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>STOP</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>STOP</td>
</tr>
<tr>
<td>41</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>STOP</td>
</tr>
<tr>
<td>42</td>
<td>7</td>
<td>7</td>
<td>1</td>
<td>STOP</td>
</tr>
</tbody>
</table>

*Could also be done from within cards.*
Appendix C

Sample Caeds Universal File For Input Data

1
151
NONE
NONE
CAEDS V3R1M1: MONITOR
21-FEB-92 09:18:13 NEVER NEVER
CAEDS V3R1M1: PRE/POST PROCESSING
21-FEB-92 09:41:04 -1 -1
164
IMETRIC ABS (ST) 2
1.000000000000000000D+00 1.000000000000000000D+00 1.000000000000000000D+00
2.731499023437B0000D+02 -1 -1 -1
164
1MON, FULL SCREEN
1 0.000000E+00 1.000000E+00 0.000000E+00 1.000000E+00 -1
-1
154
2TWO, SIDE BY SIDE
2 0.000000E+00 4.985000E-01 0.000000E+00 1.000000E+00
5.015000E-01 1.000000E+00 0.000000E+00 1.000000E+00 -1 -1
154
3THREE, LARGE LEFT
3 0.000000E+00 6.618000E-01 0.000000E+00 1.000000E+00
6.618000E-01 1.000000E+00 8.020000E-01 1.000000E+00
6.618000E-01 1.000000E+00 0.000000E+00 4.980000E-01
-1 -1
164
4FOUR, EQUAL SIZE
4 0.000000E+00 4.985000E-01 8.020000E-01 1.000000E+00
5.015000E-01 1.000000E+00 5.020000E-01 1.000000E+00
0.000000E+00 4.985000E-01 0.000000E+00 4.980000E-01
5.015000E-01 1.000000E+00 0.000000E+00 4.980000E-01
-1 -1
160
1OBSERVER
1 15 0 0 0 0
0.000000E+00 0.000000E+00 0.000000E+00 0.000000E+00
0.000000E+00 0.000000E+00 1.000000E+00 0.000000E+00
0.000000E+00 1.000000E+00 -1 -1
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>160</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2RIGHT_SHOULDER</td>
<td>0</td>
<td>16</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
</tr>
<tr>
<td></td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
<td>1.000000E+00</td>
<td>0.000000E+00</td>
</tr>
<tr>
<td></td>
<td>0.000000E+00</td>
<td>3.460000E+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3LEFT_SHOULDER</td>
<td>0</td>
<td>16</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
</tr>
<tr>
<td></td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
<td>1.000000E+00</td>
<td>0.000000E+00</td>
</tr>
<tr>
<td></td>
<td>0.000000E+00</td>
<td>3.460000E+00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>436</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|        |        |        |        |        |
|        |        |        |        |        |
| 4BLACK  | 0.000000E+00 | 0.000000E+00 | 0.000000E+00 |        |
|        | -1       | -1      |        |        |
|        | 436      |        |        |        |

|        |        |        |        |        |
|        |        |        |        |        |
| 1BLUE   | 0.000000E+00 | 0.000000E+00 | 1.000000E+00 |        |
|        | -1       | -1      |        |        |
|        | 436      |        |        |        |

|        |        |        |        |        |
|        |        |        |        |        |
| 2GRAY_BLUE | 0.000000E+00 | 3.300000E-01 | 1.000000E+00 |        |
|        | -1       | -1      |        |        |
|        | 436      |        |        |        |

|        |        |        |        |        |
|        |        |        |        |        |
| 3LIGHT_BLUE | 0.000000E+00 | 5.600000E-01 | 1.000000E+00 |        |
|        | -1       | -1      |        |        |
|        | 436      |        |        |        |

|        |        |        |        |        |
|        |        |        |        |        |
| 4CYAN   | 0.000000E+00 | 1.000000E+00 | 1.000000E+00 |        |
|        | -1       | -1      |        |        |
|        | 436      |        |        |        |

|        |        |        |        |        |
|        |        |        |        |        |
| 5DARK_GREEN | 0.000000E+00 | 3.300000E-01 | 0.000000E+00 |        |
|        | -1       | -1      |        |        |
|        | 436      |        |        |        |

|        |        |        |        |        |
|        |        |        |        |        |
| 6DARK_GREEN | 0.000000E+00 | 6.600000E-01 | 0.000000E+00 |        |
|        | -1       | -1      |        |        |
|        | 436      |        |        |        |

|        |        |        |        |        |
|        |        |        |        |        |
| 7GREEN  | 0.000000E+00 | 1.000000E+00 | 0.000000E+00 |        |
|        | -1       | -1      |        |        |
|        | 436      |        |        |        |

|        |        |        |        |        |
|        |        |        |        |        |
| 8YELLOW | 1.000000E+00 | 1.000000E+00 | 0.000000E+00 |        |
|        | -1       | -1      |        |        |
|        | 436      |        |        |        |
9GOLDEN_ORANGE
1.000000E+00 6.600000E-01 0.000000E+00
-1
-1
436
1ORANGE
1.000000E+00 3.300000E-01 0.000000E+00
-1
-1
436
1RED
1.000000E+00 0.000000E+00 0.000000E+00
-1
-1
436
12MAGENTA
1.000000E+00 0.000000E+00 1.000000E+00
-1
-1
436
13LIGHT_MAGENTA
1.000000E+00 3.300000E-01 1.000000E+00
-1
-1
436
14PINK
1.000000E+00 6.600000E-01 1.000000E+00
-1
-1
436
15WHITE
1.000000E+00 1.000000E+00 1.000000E+00
-1
-1
800
1
WORKING_SET1
-1
-1
801
1 0 1 8 1 1 0 0
0 0 0 1 0
1 1
0.000000000000000D+00 0.000000000000000D+00 0.000000000000000D+00
2.900000000000000D+02 0.000000000000000D+00 0.000000000000000D+00
2 0 1 8 1 1 0 0
0 0 0 1 0
1 1
2.900000000000000D+02 0.000000000000000D+00 0.000000000000000D+00
1.550000000000000D+03 0.000000000000000D+00 0.000000000000000D+00
3 0 1 8 1 1 0 0
0 0 0 1 0
1 1
1.550000000000000D+03 0.000000000000000D+00 0.000000000000000D+00
2.060000000000000D+03 0.000000000000000D+00 0.000000000000000D+00
4 0 1 8 1 1 0 0
0 0 0 1 0
1 1
2.060000000000000D+03 0.000000000000000D+00 0.000000000000000D+00
2.060000000000000D+03 2.000000000000000D+02 0.000000000000000D+00
6 0 1 8 1 1 0 0
0 0 0 1 0
1 1
<table>
<thead>
<tr>
<th>2.0500000000000000D+03</th>
<th>2.0000000000000000D+02</th>
<th>0.0000000000000000D+00</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5500000000000000D+03</td>
<td>2.0000000000000000D+02</td>
<td>0.0000000000000000D+00</td>
</tr>
<tr>
<td>6 0 1 8 1 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6500000000000000D+03</td>
<td>2.0000000000000000D+02</td>
<td>0.0000000000000000D+00</td>
</tr>
<tr>
<td>7 0 1 8 1 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.9000000000000000D+02</td>
<td>2.0000000000000000D+02</td>
<td>0.0000000000000000D+00</td>
</tr>
<tr>
<td>8 0 1 8 1 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0000000000000000D+00</td>
<td>2.0000000000000000D+02</td>
<td>0.0000000000000000D+00</td>
</tr>
<tr>
<td>9 0 1 8 1 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.9000000000000000D+02</td>
<td>2.0000000000000000D+02</td>
<td>0.0000000000000000D+00</td>
</tr>
<tr>
<td>10 0 1 8 1 1 0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.9500000000000000D+02</td>
<td>2.0000000000000000D+02</td>
<td>0.0000000000000000D+00</td>
</tr>
<tr>
<td>1 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.6500000000000000D+03</td>
<td>2.0000000000000000D+02</td>
<td>0.0000000000000000D+00</td>
</tr>
<tr>
<td>1 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PHYSICAL PROPERTY TABLE

<table>
<thead>
<tr>
<th>1.1000000E+00</th>
<th>0.0000000E+00</th>
<th>0.0000000E+00</th>
<th>0.0000000E+00</th>
<th>1.0000000E-36</th>
<th>1.0000000E-36</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0000000E+00</td>
<td>1.0000000E-36</td>
<td>1.0000000E-36</td>
<td>0.0000000E+00</td>
<td>0.0000000E+00</td>
<td>0.0000000E+00</td>
</tr>
<tr>
<td>0.0000000E+00</td>
<td>0.0000000E+00</td>
<td>0.0000000E+00</td>
<td>0.0000000E+00</td>
<td>0.0000000E+00</td>
<td>0.0000000E+00</td>
</tr>
<tr>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>2.06800E+11</td>
<td>2.90000E-01</td>
<td>1.00000E-35</td>
<td>3.61000E-06</td>
<td>1.00000E-35</td>
</tr>
<tr>
<td>1.00000E-38</td>
<td>4.60000E+01</td>
<td>1.00000E-36</td>
<td>0.00000E+00</td>
<td>2.48000E+08</td>
</tr>
<tr>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
</tr>
<tr>
<td>1.00000E+04</td>
<td>1.60000E+09</td>
<td>1.50000E+09</td>
<td>1.60000E+00</td>
<td></td>
</tr>
<tr>
<td>1.50000E+09</td>
<td>6.80000E+07</td>
<td>0.00000E+00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

-1

-1

92

1 2 2 4 1
2 2 1 1 2 2
1 1 0 0 8 0
0.00000E+00 2.00000E+02 0.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 1.00000E+00 2.90000E+02 2.00000E+02 0.00000E+00 1.00000E+00
2.90000E+02 0.00000E+00 0.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00
1.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00 1.00000E+00 1.00000E+00
2.00000E+02 2.00000E+02 0.00000E+00 1.00000E+00 2.90000E+02 0.00000E+00
0.00000E+00 1.00000E+00 1.55000E+03 2.00000E+02 0.00000E+00 1.00000E+00
1.55000E+03 0.00000E+00 0.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00
1.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00 1.00000E+00 1.00000E+00
3 2 2 4 1
2 2 1 1 2 2
1 1 0 0 8 0
1.55000E+03 2.00000E+02 0.00000E+00 1.00000E+00 1.55000E+03 0.00000E+00
0.00000E+00 1.00000E+00 2.05000E+03 2.00000E+02 0.00000E+00 1.00000E+00
2.05000E+03 1.00000E+00 1.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00
1.00000E+00 1.00000E+00 0.00000E+00 0.00000E+00 1.00000E+00 1.00000E+00
-1

-1

743

1 -1 1 0.00000E+00
1
2 -1 2 0.00000E+00
1
3 -1 2 0.00000E+00
2
4 -1 1 0.00000E+00
3
5 -1 1 0.00000E+00
3
6 -1 2 0.00000E+00
2
7 -1 2 0.00000E+00
1
8 -1 1 0.00000E+00
1

-1

744

1 1 3 1 1
1.65000E+02 0.00000E+00 0.00000E+00
1
2 2 3 1 1
9.20000E+02 0.00000E+00 0.00000E+00
2
3 3 4 1 1
1.80E+03 0.0000E+00 0.0000E+00
3 4 4 5 1 1
2.05E+03 1.0000E+02 0.0000E+00
3 5 8 6 1 1
1.80E+03 2.0000E+02 0.0000E+00
3 6 6 7 1 1
9.20E+02 2.0000E+02 0.0000E+00
2 7 7 8 1 1
1.45E+02 2.0000E+02 0.0000E+00
1 8 8 1 1 1
0.0000E+00 1.0000E+02 0.0000E+00
1 9 2 7 2 1
2.90E+02 1.0000E+02 0.0000E+00
1 2 10 3 6 2 1
1.85E+03 1.0000E+02 0.0000E+00
2 3 -1
-1
7.61
2.54E+02
-1
705
1 12 1 0 0 11 2 4
95 1 1 7 1 1 1 0
8 1 9 7
1
2 12 1 0 0 11 2 4
95 1 1 7 1 1 2 0
9 2 10 6
-1
3 12 1 0 0 11 2 4
95 1 1 7 1 1 3 0
10 3 4 5
-1
-1
15
1 0 0 11 0.0000E+00 2.0000E+02 0.0000E+00
2 0 0 11 0.0000E+00 1.0000E+02 0.0000E+00
3 0 0 11 0.0000E+00 0.0000E+00 0.0000E+00
4 0 0 11 1.45E+02 2.0000E+02 0.0000E+00
5 0 0 11 1.45E+02 0.0000E+00 0.0000E+00
6 0 0 11 2.90E+02 2.0000E+02 0.0000E+00
7 0 0 11 2.90E+02 1.0000E+02 0.0000E+00
8 0 0 11 2.90E+02 0.0000E+00 0.0000E+00
9 0 0 11 4.47E+02 2.0000E+02 0.0000E+00
10 0 0 11 4.47E+02 0.0000E+00 0.0000E+00
11 0 0 11 6.05E+02 2.0000E+02 0.0000E+00
12 0 0 11 6.05E+02 1.0000E+02 0.0000E+00
13 0 0 11 6.05E+02 0.0000E+00 0.0000E+00
14 0 0 11 7.62E+02 2.0000E+02 0.0000E+00
15 0 0 11 7.62E+02 0.0000E+00 0.0000E+00
16 0 0 11 9.20E+02 2.0000E+02 0.0000E+00
17 0 0 11 9.20E+02 1.0000E+02 0.0000E+00
18 0 0 11 9.20E+02 0.0000E+00 0.0000E+00

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
<td>11 1.077E+03 2.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
<td>11 1.077E+03 0.000E+00 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>0</td>
<td>11 1.235E+03 2.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>0</td>
<td>11 1.235E+03 1.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>0</td>
<td>0</td>
<td>11 1.235E+03 0.000E+00 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>0</td>
<td>11 1.392E+03 2.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>0</td>
<td>11 1.392E+03 0.000E+00 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>0</td>
<td>11 1.560E+03 2.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td>0</td>
<td>11 1.560E+03 1.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>0</td>
<td>11 1.560E+03 0.000E+00 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>0</td>
<td>11 1.675E+03 2.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
<td>11 1.675E+03 0.000E+00 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td>0</td>
<td>11 1.800E+03 2.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>0</td>
<td>11 1.800E+03 1.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>0</td>
<td>0</td>
<td>11 1.800E+03 0.000E+00 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>0</td>
<td>11 1.925E+03 2.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>0</td>
<td>0</td>
<td>11 1.925E+03 0.000E+00 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>0</td>
<td>0</td>
<td>11 2.000E+03 2.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>0</td>
<td>0</td>
<td>11 2.000E+03 1.000E+02 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>0</td>
<td>0</td>
<td>11 2.000E+03 0.000E+00 0.000E+00</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>0</td>
<td>0</td>
<td>11 0.000E+00 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>0</td>
<td>11 0.000E+00 1.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>0</td>
<td>0</td>
<td>11 0.000E+00 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>0</td>
<td>0</td>
<td>11 1.450E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>0</td>
<td>0</td>
<td>11 1.450E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>0</td>
<td>0</td>
<td>11 2.900E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td>0</td>
<td>11 2.900E+03 1.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>0</td>
<td>0</td>
<td>11 2.900E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>0</td>
<td>0</td>
<td>11 4.475E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>0</td>
<td>0</td>
<td>11 4.475E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>0</td>
<td>0</td>
<td>11 6.050E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0</td>
<td>11 6.050E+03 1.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>0</td>
<td>0</td>
<td>11 6.050E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>0</td>
<td>0</td>
<td>11 7.625E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>0</td>
<td>0</td>
<td>11 7.625E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>0</td>
<td>0</td>
<td>11 9.200E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>0</td>
<td>0</td>
<td>11 9.200E+03 1.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>0</td>
<td>0</td>
<td>11 9.200E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>0</td>
<td>0</td>
<td>11 1.077E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>0</td>
<td>0</td>
<td>11 1.077E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>0</td>
<td>0</td>
<td>11 1.235E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>0</td>
<td>11 1.235E+03 1.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>0</td>
<td>0</td>
<td>11 1.235E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>0</td>
<td>0</td>
<td>11 1.392E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>0</td>
<td>0</td>
<td>11 1.392E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>0</td>
<td>0</td>
<td>11 1.560E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>0</td>
<td>0</td>
<td>11 1.560E+03 1.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>0</td>
<td>0</td>
<td>11 1.560E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>0</td>
<td>0</td>
<td>11 1.675E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>0</td>
<td>0</td>
<td>11 1.675E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>0</td>
<td>0</td>
<td>11 1.800E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>0</td>
<td>0</td>
<td>11 1.800E+03 1.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>0</td>
<td>0</td>
<td>11 1.800E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>0</td>
<td>0</td>
<td>11 1.925E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>0</td>
<td>0</td>
<td>11 1.925E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>0</td>
<td>0</td>
<td>11 2.050E+03 2.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>0</td>
<td>0</td>
<td>11 2.050E+03 1.000E+02 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>0</td>
<td>0</td>
<td>11 2.050E+03 0.000E+00 6.900E+01</td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>0</td>
<td>0</td>
<td>11 0.000E+00 2.000E+02 1.470E+02</td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>0</td>
<td>0</td>
<td>11 0.000E+00 1.000E+02 1.470E+02</td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>0</td>
<td>0</td>
<td>11 0.000E+00 0.000E+00 1.470E+02</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>0</td>
<td>0</td>
<td>11 1.450E+02 2.000E+02 1.470E+02</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>01</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.450000E+02</td>
</tr>
<tr>
<td>02</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.900000E+02</td>
</tr>
<tr>
<td>03</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.900000E+02</td>
</tr>
<tr>
<td>04</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.900000E+02</td>
</tr>
<tr>
<td>05</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>4.475000E+02</td>
</tr>
<tr>
<td>06</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>4.475000E+02</td>
</tr>
<tr>
<td>07</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>6.060000E+02</td>
</tr>
<tr>
<td>08</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>6.060000E+02</td>
</tr>
<tr>
<td>09</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>6.060000E+02</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>7.625000E+02</td>
</tr>
<tr>
<td>11</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>7.625000E+02</td>
</tr>
<tr>
<td>12</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>9.200000E+02</td>
</tr>
<tr>
<td>13</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>9.200000E+02</td>
</tr>
<tr>
<td>14</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>9.200000E+02</td>
</tr>
<tr>
<td>15</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.077500E+03</td>
</tr>
<tr>
<td>16</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.077500E+03</td>
</tr>
<tr>
<td>17</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.236000E+03</td>
</tr>
<tr>
<td>18</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.236000E+03</td>
</tr>
<tr>
<td>19</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.236000E+03</td>
</tr>
<tr>
<td>20</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.392600E+03</td>
</tr>
<tr>
<td>21</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.392600E+03</td>
</tr>
<tr>
<td>22</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.392600E+03</td>
</tr>
<tr>
<td>23</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.550000E+03</td>
</tr>
<tr>
<td>24</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.550000E+03</td>
</tr>
<tr>
<td>25</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.550000E+03</td>
</tr>
<tr>
<td>26</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.707600E+03</td>
</tr>
<tr>
<td>27</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.707600E+03</td>
</tr>
<tr>
<td>28</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.707600E+03</td>
</tr>
<tr>
<td>29</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.864000E+03</td>
</tr>
<tr>
<td>30</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.864000E+03</td>
</tr>
<tr>
<td>31</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>1.864000E+03</td>
</tr>
<tr>
<td>32</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.022600E+03</td>
</tr>
<tr>
<td>33</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.022600E+03</td>
</tr>
<tr>
<td>34</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.022600E+03</td>
</tr>
<tr>
<td>35</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.181600E+03</td>
</tr>
<tr>
<td>36</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.181600E+03</td>
</tr>
<tr>
<td>37</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.181600E+03</td>
</tr>
<tr>
<td>38</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.340600E+03</td>
</tr>
<tr>
<td>39</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.340600E+03</td>
</tr>
<tr>
<td>40</td>
<td>00</td>
<td>00</td>
<td>11</td>
<td>2.340600E+03</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>19</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>24</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>26</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>27</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>28</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>31</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>35</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>37</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>38</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>39</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>41</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>42</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>43</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>44</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>45</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>46</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>47</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>48</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>49</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>51</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>52</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>53</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>54</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>55</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>56</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>57</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>58</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>59</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>60</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>61</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>62</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>63</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>64</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>65</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>66</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>67</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>68</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>69</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>70</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>71</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>72</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>73</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>74</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>75</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>76</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>77</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>78</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>79</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>81</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>82</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>83</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>84</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>85</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>86</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>87</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>88</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>89</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>90</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>91</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>92</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>93</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>94</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>95</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>96</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>97</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>98</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>99</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>101</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>102</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>103</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>104</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>105</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>106</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>107</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>108</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>109</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>110</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>111</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>112</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>113</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>114</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>115</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>116</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>117</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>118</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>119</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>120</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>121</td>
<td>6</td>
<td>95</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
0.00000E+00 0.00000E+00 0.00000E+00
36  95
0.00000E+00 0.00000E+00 0.00000E+00
39  95
0.00000E+00 0.00000E+00 0.00000E+00
40  96
0.00000E+00 0.00000E+00 0.00000E+00
41  96
0.00000E+00 0.00000E+00 0.00000E+00
42  96
0.00000E+00 0.00000E+00 0.00000E+00
-1
-1
766
1
1

RESTR AI NT S E T 1
2
6
7
8
12
17
22
27
32
36
37
38
40
44
46
50
56
60
66
69
70
74
75

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>7</th>
<th>23</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>14</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>8</td>
<td>2</td>
<td>7</td>
</tr>
</tbody>
</table>

-1
Appendix D

Program Listing

C INCLUDE (PROCESS)
PROGRAM HLRC3D
C
IMPLICIT REAL*8 (A-Z)
C . . .SWITCHES: RENUM=100:10, FORMAT=900:10
C . . .SWITCHES:
C-----------------------------------------------
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C INITIAL ERRFIL INPUT REVIEW GLOB1 DIAGEL
C LOAD CTRL1 CFILE
C
C-----------------------------------------------
COMMON/INPUTS/SHODES,EBLEM,EBDF,EBLISC,EBIT,IFLAG1,IFLAG2,IDIM,
1 EBMODE,ECOLOR,ERFEE
COMMON/INCI/FRATIC(10),NLISC(10),LOCSTB,INEPTR
COMMON/INPUTO/IFLAG3,JOINTS,IFPLOT
COMMON/CTRL1/INCREM,BIT
COMMON/SAFE1/SKG(2000000),SKGL(200000)
C COMMON/SAFE1/SKG(18000000),SKGL(18000000)
COMMON/SAFE2/R(60000),IDOF(60000),JDIAG(60000)
COMMON/IALGTM/IFLAG5
CALL INITIAL
C
----- IDUT = OUTPUT DEVICE NUMBER
C
IDUT = 13
C
----- OPEN THE ERROR MESSAGES FILE
C
CALL ERRFIL
C
----- READ THE INPUT FILE
C
CALL INPUT (IDUT, IERROR)
C
CALL REVIEW (IDOF)
C
----- DEFINE THE GLOBAL DEGREES OF FREEDOM
C
----- FIND THE BANDWIDTH AND THE LOCATION OF THE DIAGRAL TERMS
C
IN THE GLOBAL STIFFNESS MATRIX
C
IF (IFLAG3 .EQ. 0) THEN
   CALL GLOB1 (SHODES, EBDF, ETDF, IDOF)
   CALL DIAG1 (EBLEM, EBDF, ETDF, IDOF, JDIAG, ETSK, SHODES,
1   IFLAG2, IDUT)
ENDIF
C
----- ASSEMBLE THE LOAD VECTOR
C
CALL LOAD (R, IERROR)
IF (IFLAG6 .EQ. 0) CALL CTRL1 (SKG, SKGL, R, IDOF, JDIAG, ETSK,
1 BTDF, IDUT, MBAND)
100 CONTINUE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CALL CFILE
110 CONTINUE
STOP
END

C

================================== ASSEMB ==============================
C
C INCLUDE (PROCESS)
C SUBROUTINE ASSEMB ASSEMBLES THE GLOBAL STIFFNESS MATRIX AND/OR
C STORES THE NODE NUMBERS OF THE CORENT ELEMENT AND THE POSITION
C OF THE ELEMENT MATRICES IN THE GLOBAL MATRICES.
C
C CARGUMENT LIST
C
C SKG(I) = GLOBAL STIFFNESS MATRIX STORED IN A ONE DIMENSIONAL
C ARRAY USING THE SKYLINE METHOD
C R(I) = LOAD VECTOR
C IDOF(I) = VECTOR CONTAINING THE D.O.F. NUMBERS THE JOINTS
C JDIAG(I) = LOCATION OF THE DIAGONAL TERMS OF EACH COLUMN IN THE
C GLOBAL STIFFNESS MATRIX 'SKG'
C HTSK = NUMBER OF TERMS IN THE SKG MATRIX
C IFLAG2 = 0 ; SYMMETRIC STIFFNESS MATRIX
C 1 ; NONSYMMETRIC STIFFNESS MATRIX
C IERROR = ERROR CODE  GE 0 ; ERRORS DETECTED
C =0 ; NO ERRORS
C
C IMPLICIT REAL*8 (A-H,O-Z)
C
C ..SWITCHES: REHUHB=100:10, FORMAT=800:10
C ..SWITCHES:
C C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C ELINFO ELINMT LYINFO ISH3DG ISH3SL ES3DLS
C ESSHEL ZERO3SK
C
C**************************************************************************
C
C INTEGER ELNMT, ELMNT, NDATA, NTHICK, NINT, NTERM, NTERM2, NTERM3
C REAL*8 THICK
C COMMON/IPUT8/NDATA, NELM, NBDF, ELINC, MNT, IFLAG1, IFLAG2, IDIM, 
C IERROR, NTERM3, NTERM4, NTERM5
C COMMON/INCH1/FRAC(10), ELINC(10), LOCOUT, IBCPTR
C COMMON/INPUT9/THTHIC(9), IFLAG
C COMMON/INPUT2/BDF(20,6000)
C COMMON/ELSTM/EL(60,60)
C COMMON/ASSM1/I2(120)
C COMMON/ASSM2/I2(120)
C COMMON/MPC5/CORE5P(40000), ALAMB(40000), MPCDOF(40000),
C MPOADR(2,6000), EMPC, MPPST, MAPP
C COMMON/BLOC/BEIST(27,70,15), CRACK(27,70,15), INHISTY, INHIST1
C Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
$\text{IHIST2}$

COMMON/LAYERS,LTHICK(9),ZS(9),DCS(3,3),SLAYERS,MATL,LYUM
DIMENSION IDOF(1),JDIAG(1),SKG(1),SKGL(1),R(1),U(1)
DIMENSION DUSR(60,60)
IF (IFLAG2 .EQ. 0) THEN
C=VDIR: PREFER VECTOR
DO 100 K1 = 1, BTSK
    SKG(K1) = 0.
100 CONTINUE
ELSE
C=VDIR: PREFER VECTOR
DO 110 K1 = 1, BTSK/2
    SKG(K1) = 0.
    SKGL(K1) = 0.
110 CONTINUE
ENDIF
C
THE STIFFNESS AND LOAD VECTORS ARE MODIFIED TO ENFORCE CONSTRAINT
C RELATIONS EXISTING BETWEEN DIFFERENT DOFS.
C
IF (NMPC .GT. 0) THEN
    HDF = BBDF-BBDES
    DO 120 K1 = 1, NMPC
        R(IDOF(K1+MDF)) = 0.0
120 CONTINUE
    DO 140 K1 = 1, BMPC
        BDOF = IDOF(HDF+K1)
        ILOC = MPCADR(1,K1)
        LOCD = JDIAG(BDOF)
        DO 130 K2 = ILOC, ILOC + MPCADR(2,K1) - 1
            ID = HPCD0F(K2)
            IF (ID .GT. 0) THEN
                JDOF = IDOF(ID)
                IF (JDOF .GT. 0) THEN
                    IF (IFLAG2 .EQ. 0) THEN
                        LOCA = LOCD + NDOF - JDOF
                    ELSE IF (IFLAG2 .EQ. 1) THEN
                        LOCA = LOCD - NDOF + JDOF
                        SKG(LOCA) = C0EFMP(K2)
                        SKGL(LOCA) = C0EFMP(K2)
                    ENDIF
                    ELSE IF (JDOF .LT. 0) THEN
                        R(NDOF) = R(BDOF) - C0EFHP(K2)*U(ID)
                    ELSE IF (ID .EQ. 0) THEN
                        R(BDOF) = R(NDOF) - C0EFMP(K2)
                    ENDIF
130 CONTINUE
140 CONTINUE
ENDIF
C
C NCB = NUMBER OF COLUMNS IN THE B MATRIX.
C SBR = NUMBER OF ROWS IN THE B MATRIX.
C NSW = NUMBER OF NODES IN THE ELEMENT.
C MBAN = FULL BANDWIDTH OF THE STIFFNESS MATRIX
C
NCB = MBAN/2 - 1
MBEFD = 0
REWIND IHISTY
REWIND IHIST1
DO 280 ELM = 1, NELEM

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CALL ELHFO (ELUM, ITYPE, NHEL, IFLAG, ISTART, LIBES)
CALL ELIHM (ELUM, IDENT, IDTOD, NIPI, NIPI, NIPSI, 1
MATUM, THICK)
DO IBG = 1, 60
DO IMG = 1, 60
DUSK(IBG, IMG) = 0.0
END DO
END DO

C*VDIR: PREFER SCALAR

DO 180 K1 = 1, NHEL
   I1 = NBDF*(K1-1)
   I2 = NBDF*(NBDF(K1,ELUM)-1)
180 CONTINUE

C*VDIR: PREFER SCALAR

DO 170 K2 = 1, NBDF
   K = I1 + K2
   II(K) = I2 + K2
170 CONTINUE

C -- LAYERS HAVE TO BE STORED PROPERLY FOR EACH ELEMENT.

CALL LYIHFO (LYBUM, LTHICK, ZS, MATRL, DCS, HHEL, ELBUH, 1
BIPXI, HIPETA, BIPSI)

C --- THE D MATRIX OF CONCRETE/STEEL LAYER IS RECOVERED HERE.

NIP = BIPXI*HIPETA*BIPSI
IF (HATYPE(HATBUM) .EQ. 4) THEN
   READ (IHISTY) ELBUHB, LYBUMB, HGAUS, ((CRACK(L,K,LYBUM
     1),K = 1,250), L = 1, BIP)
   IF (ELHUM .LT. ELBUHB .OR. LYDUH.EE.LYBUMB .OR. BIP .LT.
     1 'BGAUS', BGAUS)
      WRITE (*, *) 'READ ERROR READING IHISTY IB ASSEMBLY'
      WRITE (*, *) 'ELBUHB', ELBUH, 'ELBUMB', ELBUMB, 'BIP', BIP,
      2 'BGAUS'
      STOP
   ENDIF
ENDIF

C ----- SENDF = NUMBER OF ELEMENT MODAL DEGREES OF FREEDOM
C ----- BCB = NUMBER OF COLUMNS IN THE B MATRIX
C ----- NRB = NUMBER OF ROWS IN THE B MATRIX
C ---- ITYPE = ELEMENT TYPE
C ----- IFLAG = ADDITIONAL IDENTIFIER FOR THE ELEMENT

IF (ITYPE .LT. 0) GO TO 270
IF (ITYPE .GT. 300) THEN
   IF (IFLAG .EQ. 0) THEN
      BCB = 3*NHEL
      NRB = 6
      SENDF = 3
      CALL ISH3DG (ITYPE, NHEL, IERROR)
   ELSE IF (IFLAG .EQ. 4) THEN
      BCB = 6*NHEL
   ENDIF
ENDIF
BRB = 6
NENDF = 6
CALL ISHESL (ITYPE, NSEL, IERROR)
ENDIF
ENDIF

--- GEOMETRICALLY LINEAR PROBLEMS

IF (IFLAG1 .EQ. 0) THEN
  IF (ITYPE .GT. 300) THEN
    IF (IFLAG .EQ. 0) THEN
      CALL ESSL3DLS (ELNUM, ITYPE, NSEL, NENDF, NRB, NCB, NIP, MATNUM, IFLAG2, IOUT)
    ELSE IF (IFLAG .EQ. 4) THEN
      CALL ESSHEL (ELNUM, ITYPE, NSEL, NENDF, NRB, NCB, NIP, MATNUM, IFLAG2, IOUT)
    ENDIF
  ENDIF
  ELSE IF (IFLAG .EQ. 0) THEN
    CALL ES3DLS (ELNUM, ITYPE, NSEL, NENDF, NRB, NCB, NIP, MATNUM, IFLAG2, IOUT)
  ENDIF
ENDIF
ENDIF

DO IBG = 1, NCB
  DO IMG = 1, NCB
    DUSK(IBG,IMG) = DUSK(IBG,IMG) + SK(IBG,IMG)
  END DO
END DO

DO IBG = 1, NCB
  DO IMG = 1, NCB
    SK(IMG,IBG) = DUSK(IMG,IBG)
  END DO
END DO

--- BDIF = DIFFERENCE BETWEEN THE GLOBAL MODAL DEGREES OF FREEDOM

BDIF = NENDF - NRBDF
DO 260 K1 = 1, NCB
  ID1 = (K1-1)/NENDF
  ID1 = K1 + ID1 * BDIF
  BDIF = IDOF((II(ID1))
  IF (BDIF .GT. 0) THEN
    LOC0 = JDIAG(BDOF)
  ELSE
    LOC0 = JDIAG(BDOF)
  ENDIF
  DO 240 K2 = 1, NCB
    ID2 = (K2-1)/NENDF
    JDOF = IDOF((II(K2*ID2*BDIF))
    IF (BDIF .GE. JDOF .AND. JDOF .GT. 0) THEN
      LOC0 = LOC0 + BDIF - JDOF
      SKG(LOC0) = SKG(LOC0) + SK(K1,K2)
    ELSE
      LOC0 = LOC0 - BDIF + JDOF
      SKG(LOC0) = SKG(LOC0) + SK(K2,K1)
    ENDIF
  END DO
END IF
END IF

DO 240 K2 = 1, NCB
  ID2 = ((K2-1)/NENDF)
  JDOF = IDOF((II(K2+ID2*BDIF))
END DO

--- BDIF = DIFFERENCE BETWEEN THE ELEMENT MODAL DEGREES OF FREEDOM

260 CONTINUE
240 CONTINUE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IF (JDOF .GT. 0) R(JDOF) = R(JDOF) - SK(K1,K2)*U(I(K1,K2))

CONTINUE

ENDIF

CONTINUE

CONTINUE

CONTINUE

RETURN

END

C

================================================================================ E L S T I F =====================================================================
C
C INCLUDE (PROCESS)
C
C SUBROUTINE ELSTIF
C
C PROGR A
C
C ELSTIF EVALUATES THE STIFFNESS MATRIX OF EACH ELEMENT.
C
C ENTRY POINTS
C
C ES3DLS: FOR 3D STRAIN FIELDS WITHOUT GEOM. NONLINEARITY
C
C ESHEL: FOR 3D STRAIN FIELDS WITHOUT GEOM. NONLINEARITY
C
C ARGUMENT LIST
C
C ELEM = ELEMENT NUMBER
C
C NWELL = NUMBER OF NODES IN THE ELEMENT
C
C NIRB = NUMBER OF ROWS OF THE B MATRIX
C
C NCB = NUMBER OF COLUMNS OF THE B MATRIX
C
C HIP = TOTAL NUMBER OF INTEGRATION POINTS IN THE ELEMENT
C
C MATNUM = MATERIAL NUMBER FOR THE ELEMENT
C
C IERROR = ERROR CODE
C
C IMPLICIT REAL*8 (A-H,0-Z)
C
C...SWITCHES: RENUMB=100:10,FORHAT=900:10
C...SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ZEROSK JAC3D B3DLS MATMOD BTDB SKTRAH
C
C SHELM DIRVEC GETTHK JACHEL BSHEL SBTSHEL
C
C REAL*4 SHELL2,THICK,XYZ

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
REAL*8 X, XI, ETA, BSI, NX, NY, NZ, SI
CHARACTER*153 DUMMY
INTEGER ELSUM
COMMOR/INPUTS/ISP2B(10000)
COMMOR/INPUTS/ISHELLZ(3,10000)
COMMOR/UTIL1/STRESS(6), DUMMY
COMMOR/INPUT/THICK(9), IFLAG
COMMOR/SHLDIR/VECT(3,3,9)
COMMOR/ISHELPA/W(27)
COMMOR/TRANS/V1(3), V2(3), V3(3)
COMMOR/ELSUM/EXIT(60,60)
COMMOR/DEVICE/LDEV1, LDEV2, LDEV3, LDEV4, LDEV5, LDEV6, LDEV7, LDEV8
COMMOR/ISHELPA/W(20,27), XI(20,27), ETA(20,27), BSI(20,27), SI(27)
COMMOR/ISHELPA/W(20), XI(20), NY(20), NZ(20), SI(27)
COMMOR/INPUTS/XYZ(3,10000)
COMMOR/INPUTS/XYZ(20,27,27)
COMMOR/BSIGM/VECT(3,3,9)
COMMOR/BSIGM/VECT(20,20,20)
COMMOR/BSIGM/VECT(20,20,20)
COMMOR/BSIGM/VECT(3,10000)
COMMOR/BSIGM/EXIT(20,6000)

C ENTRY ES3DLS (ELSUM, ITYPE, BKEL, BKDF, BRB, BCB, BIP, MATSUM, IFLAG2, IOUT)
C
CALL ZEROSK (BCB)
DO 100 INTG0 = 1, BIP
   CALL JAC3D (INTG0, ELSUM, BKEL, IERROR, DETJAC)
   CALL 3DLS (BKEL)
   CALL MATMOD (ELSUM, ITYPE, MATSUM, INTG0, IFLAG, IOUT, DETJAC)
   1       CST = DETJAC*W(INTG0)
   CALL BTDB (IFLAG2, BRB, BCB, CST)
100 CONTINUE
C ENTRY ESSHEL (ELSUM, ITYPE, BKEL, BKDF, BRB, BCB, BIP, MATSUM, IFLAG2, IOUT)
C
--- EVALUATE THE LOCAL SHELL COORDINATES OF THE NODES

CALL ZEROSK (BCB)
DO 110 KI = 1, BKEL
   KP = EOP(KI,ELSUM)
   ICODE = IABD(ISP2B(KP), 4)
   IF (ICODE .GT. 0) THEN
      CALL SHEHORM (ELSUM, KI, BKEL, ITYPE, VECT(1.1,KI), VECT(1.2,KI))
   ELSE
      VECT(1.3,KI) = DBLE(SHEH2(1,KI))
      VECT(2.3,KI) = DBLE(SHEH2(2,KI))
      VECT(3.3,KI) = DBLE(SHEH2(3,KI))
      CALL DIREC (VECT(1.1,KI), VECT(1.2,KI), VECT(1.3,KI))
   ENDIF
110 CONTINUE
C CALL ZEROSK (BCB)
DO 120 INTG0 = 1, BIP
   CALL GETTHK (INTG0, ELSUM, BKEL, THICK, RAD)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
vector V3 returned by JACSHL is normal to mid surface of the shell at integration points

ISET = 0
SIP = SI(INTGPH)
CALL JACSHL(INTGPH, ELNUM, NNEL, THICKE, DETJAC, V3, ISET, SIP)

THE COORDINATES VECTORS V1 AND V2 ARE EVALUATED BY DIRVEC

CALL DIRVEC (V1, V2, V3)

CALL BSHL (ELNUM, NNEL, INTGPH, THICKE)
CALL MATHOD (ELNUM, ITYPE, MATNUM, INTGPH, IFLAG, IOUT, DETJAC
1 , 0, NHEL)
CST = DETJAC*W(INTGPH)
CALL BTDB (IFLAG2, NRB, NCB, CST)

CDHTIHUE
CALL SKTSHL (ELNUM, NHEL, 6, NCB, 3, 2)
CALL SKTRAH (ELNUM, NHEL, 6, NCB, 3, 3)

RETURN
END

SUBROUTINE ZEROSK(H)

C INCLUDE (PROCESS)

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C

REAL*8 SK
COMMON/ELST1/SK(60,60)

DO 110 K1 = 1, H
  DO 100 K2 = 1, H
    SK(K1,K2) = 0.
100 CONTINUE
110 CONTINUE

RETURN
END

SUBROUTINE SKTRAH (ELNUM, NHEL, NEDF, NCB, IDIM, ICASE)

C INCLUDE (PROCESS)

SUBROUTINE SKTRAN(ELNUM,NHEL,NEDF,NCB,IDIM,ICASE)

C PROGRAM:
C SKTRAH MODIFIES THE ELEMENT STIFFNESS MATRIX FOR THE SKW
BOUNDARY CONDITIONS USING TRANSFORMATION.

ARGUMENT LIST:

ELN = element number
NBEL = number of nodes in the element
NMDF = number of element nodal degrees of freedom
NCB = number of columns of the B matrix
IDIM = physical dimension of the problem (i.e., 2D or 3D)
ICASE = special code

1; transform displacement components
2; transform rotation components
3; transform all components

IMPLICIT REAL*8 (A-H,O-Z)

SWITCHES: REDUMB=100:10, FORHAT=90:10

SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

DIRCOS

INTEGER ELN
COMMON/ELST1/SK(60,60)
COMMON/TRANS/DC(3,3)
COMMON/SELDIR/VECT(3,3,9)
COMMON/B1/B1(6,60)
COMMON/ISP/ISP9(10000)
COMMON/ISP2/ISP2(20,6000)
DIMENSION CST(60,3)

IS = 0
IE = 0
IF (ICASE.EQ.1) THEN
    IE = 1
    IS = 1
ELSE IF (ICASE.EQ.2) THEN
    IS = 2
ELSE IF (ICASE.EQ.3) THEN
    IS = 1
    IE = 2
ENDDF

DO 190 K1 = 1, NBEL
    BODE = BOP(K1,ELN)
    DO 180 IC = IS, IE
        ICODE = IAND(ISP(BODE),IC)
        IF (ICODE.GT.0) THEN
            CALL DIRCOS (ICODE, IDIM, BODE)
            I = NMDF*(K1-1) + IDIM*(IC-1)
    ENDIF
    DO 130 K2 = 1, NCB
        DO 110 K3 = 1, IDIM
            CST(I,K2,K3) = 0.
        ENDIF
        DO 100 IDIR = 1, IDIM
            ID = I + IDIR
            }
C*VDIR: PREFER VECTOR

DO 130 K2 = 1, NCB
    DO 110 K3 = 1, IDIM
        CST(K2,K3) = 0.
    ENDIF
    DO 100 IDIR = 1, IDIM
        ID = I + IDIR
    ENDIF
CST(K2,K3) = CST(K2,K3) + SK(K2,ID)*DC(IDIR,K3)
1
100 CONTINUE
110 CONTINUE
C
DO 120 K3 = 1, IDIM
   ID = I + K3
   SK(K2,ID) = CST(K2,K3)
120 CONTINUE
130 CONTINUE
C
C*VDIR: PREFER VECTOR
   DO 170 K2 = 1, NCB
      DO 160 K3 = 1, IDIM
         CST(K2,K3) = 0.
      C
      DO 140 IDIR = 1, IDIM
         ID = I + IDIR
         CST(K2,K3) = CST(K2,K3) + SK(ID,K2)*DC(IDIR,K3)
140 CONTINUE
150 CONTINUE
C
DO 160 K3 = 1, IDIM
   ID = I + K3
   SK(ID,K2) = CST(K2,K3)
160 CONTINUE
170 CONTINUE
ENDIF
180 CONTINUE
190 CONTINUE
RETURN
C
ENTRY SATSHL (ELSUM, HSEL, HENDF, NCB, IDIM, ICASE)
IS = 0
IE = 0
IF (ICASE .EQ. 1) THEN
   IE = 1
   IS = 1
ELSE IF (ICASE .EQ. 2) THEN
   IE = 2
   IS = 2
ELSE IF (ICASE .EQ. 3) THEN
   IE = 1
   IS = 2
ENDIF
C
DO 290 K1 = 1, HSEL
   NODE = BOP(K1,ELSUM)
   ICODE = 1ASHD(ISPB(NODE),4)
   IF (ICODE .GT. 0) THEN
      DO 280 IC = IS, IE
         I = HENDF*(K1-1) + IDIM*(IC-1)
   C
C*VDIR: PREFER VECTOR
C
   DO 230 K2 = 1, NCB
      DO 210 K3 = 1, IDIM
         CST(K2,K3) = 0.
      DO 200 IDIR = 1, IDIM
         CST(K2,K3) = CST(K2,K3) + SK(ID,K2)*DC(IDIR,K3)
200 CONTINUE
210 CONTINUE
230 CONTINUE
239
ID = I + IDIR
CST(K2,K3) = CST(K2,K3) + SK(K2,ID)*VECT(K3,IDIR,K1)
1
200 CONTINUE
210 CONTINUE

C
DO 220 K3 = 1, IDIM
ID = I + K3
SK(K2,ID) = CST(K2,K3)
220 CONTINUE

C
DO 230 K3 = 1, IDIM
ID = I + K3
SK(ID,K2) = CST(K2,K3)
230 CONTINUE

C
ENDIF
290 CONTINUE
RETURN

ENTRY BTSHL (ELNUM, NDEL, HENDF, NRB, IDIM, ICASE)
IS = 0
IE = 0
IF (ICASE .EQ. 1) THEN
IE = 1
IS = 1
ELSE IF (ICASE .EQ. 2) THEN
IE = 2
IS = 1
ELSE IF (ICASE .EQ. 3) THEN
IE = 1
IS = 2
ENDIF
300 CONTINUE

C
DO 310 K3 = 1, IDIM
CST(K2,K3) = 0.
310 CONTINUE

C
DO 320 K2 = 1, NRB
DO 330 K3 = 1, IDIM
CST(K2,K3) = 0.
DO 330 IDIR = 1, IDIM
ID = I + IDIR
CST(K2,K3) = CST(K2,K3) + R(K2, ID)*VECT(K3)
330 CONTINUE

C*VDIR: PREFER VECTOR
DO 340 IDIR = 1, IDIM
CST(K2,K3) = 0.
340 CONTINUE

C
DO 350 K2 = 1, NRB
NODE = N0P(K1, ELNUM)
ICODE = IAND(ISIPB(NODE),4)
IF (ICODE .GT. 0) THEN
DO 360 IC = IS, IE
I = NENDF*(K1-1) + IDIM*(IC-1)
360 CONTINUE

C*VDIR: PREFER VECTOR
DO 370 K2 = 1, NRB
DO 380 K3 = 1, IDIM
CST(K2,K3) = 0.
DO 380 IDIR = 1, IDIM
ID = I + IDIR
CST(K2,K3) = CST(K2,K3) + B(K2, ID)*VECT(K3)
380 CONTINUE

END
DECLARATION: 

IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RESUME=100:10,FORMAT=900:10
C...SWITCHES:
C***************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C
C SUBROUTINE DIRCOS (ICODE, IDIM, NODE)
C
C PROGRAM:
C
C DIRCOS returns the transformation matrix which is used to
C define the local coordinates at a node.
C
C ARGUMENT LIST
C
C ICODE = 1; get the transformation matrix for the
C translational local coordinate.
C
C ICODE = 2; get the transformation matrix for the
C rotational local coordinate.
C
C IDIM = physical dimension of the problem (i.e., 2D or 3D)
C
C NODE = node number of the structure
C
C COMMON BLOCKS
C
C XAXIS = contains the first two direction cosines of the
C translation and rotation X axis.
C
C YAXIS = contains the first two direction cosines of the
C translation and rotation Y axis.
C
C DC(I,J) = transformation matrix which has its columns equal
C to the direction cosines of the local X, Y and Z axes
C
C***************************************************************

C DO 320 K3 = 1, IDIM
C ID = I + K3
C B(K2,ID) = CST(K2,K3)
C
C CONTINUE
C
C END IF
C
C CONTINUE
C
C RETURN
C
C END

C***************************************************************
C INCLUDE (PROCESS)
C SUBROUTINE DIRCOS (ICODE, IDIM, NODE)
C***************************************************************
**REAL**-4 XAXIS,YAXIS

**LOGICAL**-4 TEST1,TEST2

COMMON /INPUT/XAXIS(4,10000),YAXIS(4,10000)

COMMON /INPUT/ISPB(10000)

COMMON /TRANS/D(3,3)

**C**

IF (ICODE .EQ. 1) THEN

DC(1,1) = DBLE(XAXIS(1,NODE))
DC(1,2) = DBLE(YAXIS(1,NODE))
DC(2,1) = DBLE(XAXIS(2,NODE))
DC(2,2) = DBLE(YAXIS(2,NODE))

ELSE IF (ICODE .EQ. 2) THEN

DC(1,1) = DBLE(XAXIS(3,NODE))
DC(1,2) = DBLE(XAXIS(4,NODE))
DC(2,1) = DBLE(YAXIS(3,NODE))
DC(2,2) = DBLE(YAXIS(4,NODE))

ENDIF

IF (IDIM .EQ. 2) THEN

RETURN

ELSE IF (IDIM .EQ. 3) THEN

---- find the third direction cosine of the local X and Y axes.

DC(3,1) = DSQRT(1.0D0-DC(1,1)**2+DC(2,1)**2)
DC(3,2) = DSQRT(1.0D0-DC(1,2)**2+DC(2,2)**2)

---- Check the sign bits for the third direction cosines.

TEST1 = .FALSE.
TEST2 = .FALSE.

IF (ICODE .EQ. 1) THEN

TEST1 = BTEST(ISPB(NODE),3)
TEST2 = BTEST(ISPB(NODE),4)

ELSE IF (ICODE .EQ. 2) THEN

TEST1 = BTEST(ISPB(NODE),5)
TEST2 = BTEST(ISPB(NODE),6)

ENDIF

IF (TEST1) DC(3,1) = -DC(3,1)

IF (TEST2) DC(3,2) = -DC(3,2)

---- take the cross product of the first two vectors to find the

---- the local Z axis.

CALL CROSS (DC(1,1),DC(1,2),DC(1,3))

ENDIF

RETURN

END

====BTDB====

SUBROUTINE BTDB(IFLAG2,HRB,DBG,CST)

---BTDB evaluates B Trans CST, WHERE

---BT = TRANSPOSE OF THE B MATRIX

---CST = MATERIAL STIFFNESS MATRIX...
C CST = CONSTANT VALUE TO BE MULT. WITH EACH TERM OF
C RESULTING MATRIX.
C
C ARGUMENT LIST
C
C IFLAG2 = 0; FOR SYMMETRIC STIFFNESS MATRIX
C 1; FOR NONSYMMETRIC STIFFNESS MATRIX
C
C NR = NUMBER OF ROWS IN THE B MATRIX
C NC = NUMBER OF COLUMNS IN THE B MATRIX
C
C CST = INTEGRATION CONSTANT
C
C
C IMPLICIT REAL*8(A-H,D-2)
C
C...SWITCHES: RESUME=100:10,FORMAT=900:10
C...SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C***************************************************************************
C
REAL*8 SK,B,DEP,CST,DUMMY,TEMP
COMMON/ELST1/SK(60,60)
COMMON/B1/B(6,60)
COMMON/MATER1/DEP(6,6)
DIMENSION DUMMY(60,6)
C
C B(K3,K1) IS THE TRANSPOSE OF THE B(K1,K3)
C
DO 110 K1 = 1, NR
DO 100 K2 = 1, NR
DEP(K1,K2) = DEP(K1,K2)*CST
100 CONTINUE
110 CONTINUE
C
IF (IFLAG2 .EQ. 0) THEN
C
C*VDIR: PREFER VECTOR
C
DO 140 K1 = 1, NC
DO 130 K2 = 1, NR
TEMP = 0.
DO 120 K3 = 1, NR
TEMP = TEMP + B(K3,K1)*DEP(K3,K2)
120 CONTINUE
DUMMY(K1,K2) = TEMP
130 CONTINUE
140 CONTINUE
C
C*VDIR: PREFER VECTOR
C
DO 170 K1 = 1, NC
DO 160 K2 = 1, NC
TEMP = 0.
DO 150 K3 = 1, NR
TEMP = TEMP + DUMMY(K1,K3)*B(K3,K2)
150 CONTINUE
160 CONTINUE
170 CONTINUE
C
C***************************************************************************
CONTINUE
SK(K1,K2) = SK(K1,K2) + TEMP
CONTINUE
CONTINUE
ELSE IF (IFLAG2 .EQ. 1) THEN
C*VDIR: PREFER VECTOR
DO 220 K1 = 1, NCB
    DO 190 K2 = 1, NRB
        DUMMY(K1,K2) = 0.
        DO 180 K3 = 1, NRB
            DUMMY(K1,K2) = DUMMY(K1,K2) + B(K3,K1)*DEP(K3,K2)
        CONTINUE
    CONTINUE
DO 210 K4 = 1, NCB
    DO 200 K5 = 1, NRB
        SK(K1,K4) = SK(K1,K4) + DUMMY(K1,K5)*B(K5,K4)
    CONTINUE
CONTINUE
CONTINUE
ENDIF
RETURN
END
C
C SUBROUTINE B2D3D
C SUBROUTINE B2D3D EVALUATES THE 'B' MATRIX FOR THE 2D AND 3D
C FINITE STRAIN PROBLEMS.
C ENTRY POINTS:
C B3DLS : FOR 3D STRAIN FIELDS WITHOUT GEOMETRIC NONLINEARITY
C BSHLS : FOR 3D STRAIN FIELDS WITHOUT GEOMETRIC NONLINEARITY
C FOR SHELLS
C B(I,J) = VARIATIONAL STRAIN-DISPLACEMENT STIFFNESS
C MATRIX.
C
C
C ***:==================== B 2 D 3 D 2 2 D 3 D :=======
C
C INCLUDE (PROCESS)
SUBROUTINE B2D3D
C
C IMPLICIT REAL*8 (A-H,N-Z)
C . . .SWITCHES: RENUMB=100:10,FORMAT=900:10
C . . .SWITCHES:
C******************************************************************************
C
C SUBROUTINES OR FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C**************************************************************
INTEGER ELNUM, NEL, NEUF, NLAYRS
REAL*8 ELTHICK, THICK
COMMON/INPUT/THICK(N), IFLAG
COMMON/MAIN2/UTOTAL(60000)
C
ENTRY B3DLS(NNEL)
C
C — CALCULATION OF THE B MATRIX
C
C*VDIR: ASSUME CQUHT(20)
DO 100 K1 = 1, NHEL
     K13 = 3*K1
     K12 = K13 - 1
     K11 = K13 - 2
     TEMPI = NX(K1)
     B(1,K11) = TEMPI
     B(4,K12) = TEMPI
     B(6,K13) = TEMPI
     TEMP1 = NY(K1)
     B(2,K11) = TEMP1
     B(5,K12) = TEMP1
     B(8,K13) = TEMP1
     TEMPS = NZ(K1)
     B(3,K11) = TEMPS
     B(6,K12) = TEMPS
     B(9,K13) = TEMPS
     B(1,K12) = ZERO
     B(4,K13) = ZERO
     B(7,K11) = ZERO
     B(3,K12) = ZERO
     B(6,K13) = ZERO
     B(2,K11) = ZERO
     B(3,K12) = ZERO
     B(4,K13) = ZERO
     B(5,K11) = ZERO
     B(8,K12) = ZERO
     B(9,K13) = ZERO
100 CONTINUE
C
RETURN
C
ENTRY BSHEL (ELNUM, NEL, IHTGPH, THICKE)
C
ASI = SI(IHTGPH)
CONST = 0.5*THICKE
DO 110 K1 = 1, NHEL
C
ENTRY BSHEL (ELNUM, NEL, IHTGPH, THICKE)
C
ASI = SI(IHTGPH)
CONST = 0.5*THICKE
DO 110 K1 = 1, NHEL
C
C — calculate the B matrix for shell elements
C
C  get the local shell coordinates of the node
C
V1(1) = VECT(1,1,K1)
\[ V_2(1) = \text{VECT}(1,2,K1) \]
\[ V_3(1) = \text{VECT}(1,3,K1) \]
\[ V_1(2) = \text{VECT}(2,1,K1) \]
\[ V_2(2) = \text{VECT}(2,2,K1) \]
\[ V_3(2) = \text{VECT}(2,3,K1) \]
\[ V_1(3) = \text{VECT}(3,1,K1) \]
\[ V_2(3) = \text{VECT}(3,2,K1) \]
\[ V_3(3) = \text{VECT}(3,3,K1) \]

C: ----- compute the B matrix

\[ \text{HSIX} = S(K1,\text{INTGPP})\ast \text{SIX} \]
\[ \text{HSIY} = H(K1,\text{INTGPP})\ast \text{SIY} \]
\[ \text{HSIZ} = S(K1,\text{INTGPP})\ast \text{SIZ} \]
\[ \text{HXASI} = HX(K1)\ast \text{ASI} \]
\[ \text{HYASI} = HY(K1)\ast \text{ASI} \]
\[ \text{HZASI} = HZ(K1)\ast \text{ASI} \]

K11 = 6\ast(K1-1) + 1
K12 = K11 + 1
K13 = K12 + 1
K14 = K13 + 1
K15 = K14 + 1
K16 = K15 + 1
B(1,K11) = HX(K1)
B(2,K11) = ZERO
B(3,K11) = ZERO
B(4,K11) = HY(K1)
B(5,K11) = ZERO
B(6,K11) = HZ(K1)
B(1,K12) = ZERO
B(2,K12) = DY(K1)
B(3,K12) = ZERO
B(4,K12) = DX(K1)
B(5,K12) = ZERO
B(6,K12) = ZERO
B(1,K13) = ZERO
B(2,K13) = ZERO
B(3,K13) = HZ(K1)
B(4,K13) = ZERO
B(5,K13) = DY(K1)
B(6,K13) = HX(K1)
B(1,K14) = \text{COBST}\ast\text{KBXASI+BSIX}\ast V2(1)
B(2,K14) = \text{COBST}\ast\text{HYASI+BSIY}\ast V2(2)
B(3,K14) = \text{COBST}\ast\text{HZASI+BSIZ}\ast V2(3)
B(4,K14) = \text{COBST}\ast\text{HYASI+BSIY}\ast V2(4)
B(5,K14) = \text{COBST}\ast\text{HZASI+BSIZ}\ast V2(5)
B(6,K14) = ZERO
B(1,K15) = \text{COBST}\ast\text{KBXASI+BSIX}\ast V1(1)
B(2,K15) = \text{COBST}\ast\text{HYASI+BSIY}\ast V1(2)
B(3,K15) = \text{COBST}\ast\text{HZASI+BSIZ}\ast V1(3)
B(4,K15) = \text{COBST}\ast\text{HYASI+BSIY}\ast V1(4)
B(5,K15) = \text{COBST}\ast\text{HZASI+BSIZ}\ast V1(5)
B(6,K15) = ZERO
B(1,K16) = ZERO
B(2,K16) = ZERO
B(3,K16) = ZERO
B(4,K16) = ZERO
B(5,K16) = ZERO
B(6,K16) = ZERO

110 CONTINUE

C: RETURN
SUBROUTINE GETSTR(IGUT)

C SUBROUTINE GETSTR ASSEMBLES THE GLOBAL STIFFNESS MATRIX AND/OR
C STORES THE NODE NUMBERS OF THE CURRENT ELEMENT AND THE POSITION
C OF THE ELEMENT MATRICES IN THE GLOBAL MATRICES.

C II(J)  POSITION OF LOCAL STIFFNESS TERMS IN THE GLOBAL
C STIFFNESS MATRIX.
C
C SKG(I) = GLOBAL STIFFNESS MATRIX IN THE CONDENSED FORM
C SK(I,J) = ELEMENT STIFFNESS MATRIX
C (SK IS COMPUTED BY SUBPROGRAM STIFEL)
C
C IMPLICIT REAL*8 (A-H,O-Z)
C . . .SWITCHES: RENUMB=100:10,FORMAT=900:10
C . . .SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO  ELNTH  LINFO  ISH3DG  ISHEL  S3DL5
C $SSL
C
C******************************************************************************
C INTEGER ELHUM,ELNBMS
REAL*8 THICK
REAL*8 LTTHICK
COMM0R/INPUT8/NODES,WELEM,BDIF,ELINC1,MBIT,IFLAG1,IFLAG2,IDIM,
IVNOD0,NCOLOR,WRSE
COMM0R/IDINC1/FRACT(10),ELINC10,LDOCST,INCPTR
COMM0R/INPUT9/THICK(9),IFLAG
COMM0R/INPUT7/BIPSI,BIPETA,BIPS1,BIP,INTOD
COMM0R/INPUT2/BDF(20,50000)
COMM0R/ASSEMBL/IT(120)
COMM0R/LAYERS/LTHICK(9),ZS(9),DCS(3,3),LAYRS,MAIRL,LYHUM
COMM0R/INPUTF/MATLPE(10)
COMM0R/BLOCKS/EISTO(27,70,15),CRACK(27,250,15),IBISTY,ISTY1
$,ISTY2
COMM0R/MA194/RE(60000)
COMM0R/SAFEZ/R(60000),IDOF(60000),BJIA(60000)
COMM0R/NCPS/CDFNP(40000),LAMB(40000),MPC0DF(40000),
$ MPCADR(2,5000),NHPC,NPCST,NMXPC
C
C ----- NGB = number of columns in the B matrix
C ----- NRG = number of rows in the B matrix
C ----- NMB = number of nodes in the element
C
C REWIND IBISTY
REWIND IBIST1
DO 140 ELNUM = 1, RELEM
C
CALL ELINFO (ELNUM, ITYPE, BHEL, IFLAG, ISTART, LINES)
CALL ELINFO (ELNUM, IDEST, ISTCOD, BIPXI, BIPETA, BIPSI, MATNUM, THICK)

C
C*VDIR: PREFER SCALAR

    DO 110 K1 = 1, BHEL
     I1 = HEDDF*(K1-1)
     I2 = HEDDF*(HELDF(K1,ELNUM)-1)

C
C*VDIR: PREFER SCALAR

    DO 100 K2 = 1, HELDF
     K = I1 + K2
     II(K) = I2 + K2

100   CONTINUE

110   CONTINUE

C
DO 120 LYHUM = 1, HELAYS

    CALL LINFO (LYHUM, LTHICK, ZS, MATRL, DCS, BHEL, ELNUM, BIPXI, BIPETA, BIPSI)

C
120   NIP = BIPXI*HELDF*HEL

IF (MATYPE(MATNUM) .EQ. 4) THEN
   READ (IHISTY) ELHUMB, LYBUM, BGAUS, ((CRACK(L,K,LYHUM),K = 1,250), L = 1, NIP)
   IF (ELHUM .LT. ELHUMB .OR. LYBUM .LT. LYBUM .OR. BIP .LT. BIP .OR.
      BGAUS) THEN
      WRITE (* , **) 'READ ERROR READING IHISTY IN GETSTR'
      WRITE (* , 100) 'ELHUMB', ELHUMB, 'LYBUM', LYBUM, 'BIP', BIP, 'BGAUS', BGAUS
      STOP

ENDIF

100   STOP

C

C               NEHDF = number of element nodal degrees of freedom
C               HCB = number of columns in the B matrix
C               HRB = number of rows in the B matrix
C               ITYPE = element type
C               IFLAG = additional identifier for the element

C
IF (ITYPE .LE. 0) GO TO 130
IF (ITYPE .NE. 0 .OR. IDENT .NE. 0) THEN
   IF (ITYPE .GT. 300) THEN
      IF (IFLAG .EQ. 0) THEN
         NCB = 3*BHEL
         NRB = 6
         NEHDF = 3
         CALL ISH3DG (ITYPE, BHEL, TERROR)
      ELSE IF (IFLAG .EQ. 4) THEN
         NCB = 6*BHEL
         NRB = 6
         NEHDF = 6
         CALL ISHSH1 (ITYPE, BHEL, TERROR)
      ENDIF
   ENDIF

ENDIF

130   STOP

C

C GEOMETRICALLY LINEAR PROBLEMS

C
IF (IFLAG .EQ. 0) THEN
   IF (ITYPE .GT. 300) THEN
      IF (IFLAG .EQ. 0) THEN

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CALL S3DLS (ELNUM, ITYPE, NSEL, NRS, NCS, NIP, MATNUM, NRDIF, NRDIF, ITOT)
ELSE IF (IFLAG .EQ. 4) THEN
   CALL S3SLH (ELNUM, ITYPE, NSEL, NRS, NCS, NIP, MATNUM, NRDIF, NRDIF, ITOT)
ENDIF
ENDIF
CONTINUE
CONTINUE
CONTINUE
C  IF (NMPC .GT. 0) THEN
   DO IMPC = 1, NMPC
      ILOC = MPCADR(1,IMPC)
      ICONTENT = MPCADR(2,NMPC)
      DO ICP = ILOC + 1, ILOC + ICONTENT - 1
         IDEBT = MPCDOF(ICP)
         FACTOR = COEFHPC(ICP)
         RE(IDEBT) = RE(IDEBT) - ALAMB(IMPC)^FACTOR
      END DO
   END DO
RETURN
C    E L S T R    E L S T R    E L S T R    E L S T R    E L S T R
C    E L S T R    E L S T R    E L S T R    E L S T R    E L S T R
C    E L S T R    E L S T R    E L S T R    E L S T R    E L S T R
C INCLUDE (PROCESS)
SUBROUTINE ELSTR
C SUBPROGRAM ELSTR EVALUATES THE STIFFNESS MATIRIX OF EACH ELEM.
C ENTRY POINTS:
C S3DLS : FOR 3D STRAIN FIELDS WITHOUT GEOM. NONLINEARITY
C S3SLH : FOR 3D STRAIN FIELDS WITHOUT GEOM. NONLINEARITY
C IN SHELLS
C IMPLICIT REAL*8 (A-E, O-Z)
C...SWITCHES: RESUMB=100:10,FUMAT=900:10
C...SWITCHES:
C***********************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C JACBD3D S3DLS MATNUM EQUILB SHMORM DISVEC
C GETTHK JACSLH TSHR SHEL BTHSL
C***********************************************************************
REAL*8 B,EXI,BETA,NSI,NX,EX,EXAS,EXASI,EXASII,EXASIII,EXASIV
REAL*8 LTEICK
REAL*4 SHELLZ,THICK,XYZ
INTEGER ELNUM
COMMON/LAYERS/LTEICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
COMMON/TRANS/V1(3),V12(3),V13(3)
ENTRY S3DLS (ELNUM, ITYPE, SHEL, BRB, NIP, MATNUM, BBDF, BBDF, IOUT)

DO 110 INTGPS = 1, NIP
   CALL JAC3D (INTGPS, ELNUM, SHEL, IERROR, DETJAC)
   CALL S3DLS (SHEL)
   CST = DETJAC*W(INTGPS)

   DUDX = 0.
   DVDX = 0.
   DWDX = 0.
   DUDY = 0.
   DWDY = 0.
   DUDZ = 0.
   DWDZ = 0.

*VDIR: ASSUME COUNT(20)
DO 100 K1 = 1, NBE
   K11 = NUDP*(K1-1) + 1
   K12 = K11 + 1
   K13 = K12 + 1
   DUDX = DUDX + NX(K1)*TOTAL(II(K11))
   DVDX = DVDX + NX(K1)*TOTAL(II(K12))
   DWDX = DWDX + NX(K1)*TOTAL(II(K13))
   DUDY = DUDY + NY(K1)*TOTAL(II(K11))
   DWDY = DWDY + NY(K1)*TOTAL(II(K12))
   DUDZ = DUDZ + NZ(K1)*TOTAL(II(K13))
   DWDZ = DWDZ + NZ(K1)*TOTAL(II(K13))
CONTINUE

100 CONTINUE

   STRK(1) = DUDX
   STRK(2) = DVDX
   STRK(3) = DWDX
   STRK(4) = DUDY + DVDX
   STRK(5) = DWDY + DVDX
   STRK(6) = DUDZ + DWDZ

   CALL MATMOD (ELNUM, ITYPE, MATNUM, INTGPS, IFLAG, IOUT, DETJAC)
   CALL EQUILB (CST, NCB, BRB, BBDF, BBDF)
CONTINUE

RETURN
C
ENTRY SSHL (ELNUM, ITYPE, NBEL, NB, NCB, SIP, MTSUM, ENDF, NENDF,
1, IDIM, IOUT)

C ------ evaluate the local shell coordinates of the nodes
C
DO 120 K1 = 1, NBEL
KP = BOP(K1,ELNUM)
ICODE = IAIPD(ISPB(KP),4)
C ------ If shell rotations are assembled in the local shell coordinate
C system then retrieve the local Z-axis from storage. Else
C evaluate the normal to the shell mid plane by a call to the
C SHHORM routine.
C
IF (ICODE .GT. 0) THEN
   CALL SHHORM (ELNUM, K1, NBEL, ITYPE, VECT(1,1,K1),VECT(1,2,
1,K1),VECT(1,3,K1))
ELSE
   VECT(1,2,K1) = DBLE(SHELLZ(1,KP))
   VECT(2,2,K1) = DBLE(SHELLZ(2,KP))
   VECT(3,2,K1) = DBLE(SHELLZ(3,KP))
   CALL DIRVEC (VECT(1,1,K1),VECT(1,2,K1),VECT(1,3,K1))
ENDIF
120 CONTINUE
C
BDIF = ENDF - NENDF
DO 170 K1 = 1, NBEL
     NODE = BOP(K1,ELNUM)
     ID = ENDF*NODE(-1)
DO 150 K2 = 1, IDIM
     ID1 = ID + K2
     DOF(ID1) = UTOTAL(IKIDI))
130 CONTINUE
KID = ID + IDIM
ICODE = IAIPD(ISPB(NODE),4)
IF (ICODE .GT. 0) THEN
   DO 140 K2 = 1, IDIM
       ID2 = ID + K2
       TEMP = TEMP + VECT(K2,K2,K1)*UTOTAL(IKID2))
   CONTINUE
140 DOF(ID1) = TEMP
   CONTINUE
   ELSE
   DO 150 K2 = 1, IDIM
       ID2 = ID + K2
       DOF(ID1) = UTOTAL(IKID2))
150 CONTINUE
EHDIF
170 CONTINUE
C
DO 190 ISTGPH = 1, SIP
     CALL GETTHK (ISTGPH, ELNUM, NBEL, THICKE, RAD)
C ------ vector V13 returned by JACSL is normal to mid surface of the
C shell at integration points
C
IST = 0
     SIP = SI(ISTGPH)
     CALL JACSHL(ISTGPH,ELNUM,NBEL,THICKE,DETJAC,V13,IST,SIP)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
the coordinates vectors \( V_{11} \) and \( V_{12} \) are evaluated by \texttt{DIRVEC}

which is part of the Element Library Module

\begin{verbatim}
CALL DIRVEC (V11, V12, V13)
ENDIF
CALL BSHL (ELNUM, HHEL, IHTGPH, THICKE, ITYPE)
CALL DETJAC (HHEL)

CST = DETJAC(EEL)

DU DX = 0.
D VD X = 0.
D WD Y = 0.
D VD Y = 0.
DU DZ = 0.
D VD Z = 0.
D WD Z = 0.

C*VDIR: ASSUME COUNT(8)

ASI = SI(IHTGPH)
COHST = THICKE*0.50
DO 180 K1 = 1, HHEL

C------ get the local shell coordinates of the node

V1(1) = VECT(1,1,K1)
V2(1) = VECT(1,2,K1)
V1(2) = VECT(2,1,K1)
V2(2) = VECT(2,2,K1)
V1(3) = VECT(3,1,K1)
V2(3) = VECT(3,2,K1)

K11 = NWDF*(K1-1) + 1
K12 = K11 + 1
K13 = K12 + 1
K14 = K13 + 1
K15 = K14 + 1
NSIX = H(K1,IHTGPH)*SIX
NSIY = H(K1,IHTGPH)*SIY
NSIZ = H(K1,IHTGPH)*SIZ
NSASI = NSI(K1)*ASI
NYASI = NNY(K1)*ASI
NZASI = NZ(K1)*ASI

C

DU DX = DU DX + HX(K1)*DOF(K11) + CONST*(-V1(1)*DOF(K14)) +
1 V1(1)*DOF(K15)) > (NASTI=NSIX)
D VD X = D VD X + HX(K1)*DOF(K12) + CONST*(-V1(2)*DOF(K14)) +
1 V1(2)*DOF(K15)) > (NASTI=NSIY)
D WD Y = D WD Y + HY(K1)*DOF(K13) + CONST*(-V1(3)*DOF(K14)) +
1 V1(3)*DOF(K15)) > (NASTI=NSIZ)
D VD Y = D VD Y + HY(K1)*DOF(K12) + CONST*(-V1(2)*DOF(K14)) +
1 V1(2)*DOF(K15)) > (NYASI=NSIY)
D WD Z = D WD Z + HZ(K1)*DOF(K13) + CONST*(-V1(3)*DOF(K14)) +
1 V1(3)*DOF(K15)) > (NZASI=NSIZ)
D VD Z = D VD Z + HZ(K1)*DOF(K12) + CONST*(-V1(2)*DOF(K14)) +
1 V1(2)*DOF(K15)) > (NZASI=NSIZ)
D WD Z = D WD Z + HZ(K1)*DOF(K13) + CONST*(-V1(3)*DOF(K14)) +
1 V1(3)*DOF(K15)) > (NZASI=NSIZ)

END
\end{verbatim}
\begin{verbatim}
V1(3)*DGF(K15)*(NZASI+HSIZ)
180 CONTINUE

  STRB(5) = D W D Y  + DV DZ
  STRB(6 ) = D W D X  + DU DZ

C
CALL MATMOD (ELNUM, ITYPE, MATNUM, ISTATP, IFLAG, IDOUT, DETJAC
  1
1 1, NREL)
CALL BTSHL (ELNUM, NREL, 6, NRB, 3, 2)
CALL EQUILB (CST, MCB, NRB, NDF, NEDF)
190 CONTINUE
C
RETURN
END
C
C
C=====================================================================
C
C INCLUDE (PROCESS)

SUBROUTINE EQUILB(CST, MCB, NRB, NDF, NEDF)
IMPLICIT REAL*8(A-H,0-Z)

C . . .SWITCHES: REBUHB=100:10,FORHAT=900:10
C . . .SWITCHES:
C
C SUBROUTINES OR FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C=====================================================================

REAL*8 B,RE,STRS,CST,TEMP
COMMOS/ASSEMP/II(120)
COMM/ELSTR2/STRS(6)
COMMOS/MAIN4/RE(60000)
COMMOS/B1/B(6,60)
C DIMENSION RTEMP( 60 )
C
C*VDIR: ASSUME COUNT(18)
C*VDIR: IGNORE RECORDPS
C
D0 200 Kl = 1 , HCB
  TEMP = 0.
D0 100 K2 = 1 , NRB
  100 TEMP = TEMP + B(K2, Kl)*STRS( K2 )
  200 RTEMP( Kl ) = TEMP+CST

C
BDIF = NDFP - NENDY
D0 110 K1 = 1 , BCB
  ID1 = (K1-1)/NENDF
  ID2 = K1 + ID1*BDIF
  TEMP = 0.
  DO 100 K2 = 1, NRB
    100 TEMP = TEMP + B(K2,K1)*STRS(K2)
  110 CONTINUE

RE(I2(ID2)) = RE(I2(ID2)) + TEMP+CST

C
RETURN
END

C
\end{verbatim}
SUBROUTINE GLOBAL

SUBROUTINE GLOBAL IS USED TO MODIFY THE FINAL GLOBAL STIFFNESS MATRIX. THIS IS DONE IN ORDER TO SOLVE THE SET OF SIMULTANEOUS EQUATIONS BY THE METHOD OF MODIFICATION. THIS SUBROUTINE IS DESIGNED FOR MODIFICATION OF BANDED NONSYMMETRIC MATRICES IN THEIR CONDENSED FORM.

C********************************************************************
IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RETURN=100:10,FORMAT=900:10
C...SWITCHES:
C********************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C********************************************************************

ENTRY GLOBl(NNODES, NNDF, NTDF, IDDF)
MDOF = NNDF - NNODES
ICOUNT = 0
C*VDIR: PREFER SCALAR
DO 100 ID = 1, MDOF + NHPC
   IF (IDOF(ID) .EQ. 0) THEN
      ICOUNT = ICDUNT + 1
      IDOF(ID) = ICOUNT
   ELSE IF (IDOF(ID) .GT. 0) THEN
      IDOF(ID) = 0
   ENDIF
100 CONTINUE

NTDF = ICOUNT
C
RETURN
C
ENTRY GLOB2(NNODES, NNDF, BTDF, IDDF)
MDOF = NNDF - NNODES
ICOUNT = 0
C*VDIR: PREFER SCALAR
DO 110 ID = 1, MDOF + NHPC
   IF (IDOF(ID) .EQ. 0) THEN
      ICOUNT = ICDUNT + 1
      IDOF(ID) = ICOUNT
   ELSE IF (IDOF(ID) .GT. 0) THEN
      IDOF(ID) = 0
   ENDIF
110 CONTINUE

BTDF = ICOUNT
C
RETURN
ICOUNT = ICOUNT + 1
IDOF(ID) = ICOUNT
END

110 CONTINUE
BTDF = ICOUNT

C
RETURN
END

C
INCLUDE (PROCESS)
SUBROUTINE COORD
IMPLICIT REAL'S (A-H,D-Z)
C
C. . .SWITCHES: REHUMB=100:10,FORMAT=900:10
C. . .SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C SDNORM, DIRVEC
C
C*************************************************************************

REAL*4 XYZ, SHELLZ, THICK
REAL*8 N, XI, BETA, SI
INTEGER ELHUM

COMMON/MAIN2/UTOTAL(60000)
COMMON/INPUT2/XYZ(3,10000)
COMMON/INPUT2/NEP(20,5000)
COMMON/ISHAPE(20,27), XI(20,27), BETA(20,27), SI(20,27)
COMMON/INPUT2/THICK(9), IFLAG
COMMON/SKIN/SHELLZ(3,10000)
COMMON/ISPED/ISPB(10000)
DIMENSION V(3), V1(3), V2(3)

C
C ENTRY COORD1
C
ENTRY COORD1(ELHUM, NSEL, ISTGPS, ITYPE, X1, Y1, Z1)
X1 = 0.
Y1 = 0.
Z1 = 0.

C+VDIR: ASSUME COUNT(8)
DO 100 K = 1, NSEL
IF (IFLAG .EQ. 0) THEN
X1 = X1 + B(K, ISTGPS)*XYZ(1, HOP(K, ELHUM))
Y1 = Y1 + B(K, ISTGPS)*XYZ(2, HOP(K, ELHUM))
Z1 = Z1 + B(K, ISTGPS)*XYZ(3, HOP(K, ELHUM))
ELSE IF (IFLAG .EQ. 4) THEN
KP = HOP(K, ELHUM)
ICODE = IABS(ISPB(KP), 4)
C
C---- If shell rotations are assembled in the local shell coordinate
C system then retrieve the local Z-axis from storage. Else
C evaluate the normal to the shell mid plane by a call to the
C SDNORM routine.
C
CODST = 0.5*SI(ISTGPS)
IF (ICODE .GT. 0) THEN
CALL SDNORM(ELHUM, K, NSEL, ITYPE, V1(1), V2(1), V3(1))
ELSE
V3(1) = DBLE(SHELLZ(1, KP))
V3(2) = DBLE(SHELLZ(2, KP))
V3(3) = DBLE(SHELLZ(3, KP))
END
CALL DIRVEC(V1(1),V2(1),V3(1))
ENDIF
X1 = X1 + N(K,INTGPE)*(IYZ(1,HOP(K,ELEUM))+THICK(K)*CONST*
1  V3(1))
Y1 = Y1 + N(K,INTGPE)*(IYZ(2,HOP(K,ELEUM))+THICK(K)*CONST*
1  V3(2))
Z1 = Z1 + N(K,INTGPE)*(IYZ(3,HOP(K,ELEUM))+THICK(K)*CONST*
1  V3(3))
ENDIF
100 CONTINUE
C RETURN
C ENTRY Coord2 (ELBum, BBEL, INTGPE, RSDF, UXIP, UYIP, UZIP)
U(1) = 0.
U(2) = 0.
U(3) = 0.
C*VDIR: PREFER SCALAR
DO 120 K = 1, BBEL
C*VDIR: PREFER SCALAR
DO 110 ID = 1, RSDF
X1 = RSDF*(HOP(K,ELEUM)-1) + ID
U(ID) = U(ID) + N(K,INTGPE)*UTOTAL(K)
110 CONTINUE
120 CONTINUE
UXIP = U(1)
UYIP = U(2)
UZIP = U(3)
RETURN
END

C =============================== M A T H O D ===============================
C
C INCLUDE (PROCESS)
C SUBROUTINE MATMOD(ELBUM, ITYPE, MATNUM, INTGPF, IFLAG, IOUt, DETJAC,
# ** ICQDE, Hn1)
IMPLICIT REAL*8 (A-H.O-Z)
C . . . SWITCHES: REHUHB=10,0:10, FQRMAT=900,0:10
C . . . SWITCHES:
C***************SUBRoutines AND Functions CAllED FROM THIS ROUTINE***************
C DCOMST ERRORS
C
C*******************************************************************************
C INTEGER ELBUM
CHARACTER COMM*6,BUFFER*80,BUFF*80
COMMON/INPUT/MATYPE(10)
COMMON/COMP2/COMM,BUFFER,BUFF
C
I = MATYPE(MATNUM)
IF (I .EQ. 1) THEN
CALL DCOMST (ELBUM, ITYPE, MATNUM, INTGPF, IFLAG, IOUt, DETJAC
1 , ICQDE, BBEL)
ELSE
WRITE (BUFFER, 14) 'MATERIAL', MATNUM
CALL ERRORS (18, 1, 'MATHOD')
STOP
ENDIF

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C

RETURN
END

C============================================================================

C INCLUDE (PROCESS)

SUBROUTINE GETTHK(IHTGPH, ELHUM, HHEL, THICKE, RAD)

IMPLICIT DOUBLE PRECISION (A-H,O-Z)

C . . .SWITCHES: REHUMB=100:10,FORMAT=900:10
C . . .SWITCHES:

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT

C============================================================================

INTEGER ELMU

REAL*4 THICKE,XYZ

REAL*8 H,HXI,HETA,HSI,SI

COMMON/INPUTS/XYZ(3,10000)

COMMON/INPUT2/HOP(20,10000)

COMMON/TSBAP1/HI(20,27),HIXI(20,27),HETA(20,27),HSI(20,27),SI(27)

COMMON/INPUT3/THICK(9),IFLAG

IF (IFLAG .EQ. 3) THEN

RAD = 0.0

DO 100 K1 = 1, HHEL

RAD = RAD + B(K1,IHTGPH)*XYZ(1,HOP(K1,ELHUM))

100 CONTINUE

THICKE = 2.0*3.141562*RAD

ELSE

THICKE = 0.0

RAD = 0.0

DO 110 K1 = 1, HHEL

THICKE = THICKE + H(K1,IHTGPH)*THICK(K1)

110 CONTINUE

ENDIF

RETURN

C CHANGES 7/18

C============================================================================

C INCLUDE (PROCESS)

SUBROUTINE DIAGHL(HELEH, NNDF,BTDF, IDDF, JDIAG, HTSK,NNODES, IFLAG2,

1 lOUT)

C============================================================================

C I

C THIS PROGRAM COMPUTES THE VECTOR CONTAINING THE ADDRESSES

C OF THE DIAGONAL ELEMENTS OF THE STIFFNESS MATRIX. IT ALSO

C CALCULATES THE BANDWIDTH AND THE AVERAGE BANDWIDTH OF THE

C STIFFNESS MATRIX AND PRINTS THESE STATISTICS.

C I

C I A R G U M E N T L I S T

C I

C I BELEM = TOTAL NUMBER OF ELEMENTS

C I NNDF = NUMBER OF NODAL DEGREES OF FREEDOM

C I BTDF = NUMBER OF TOTAL DEGREES OF FREEDOM

C I IDOF(I) = VECTOR CONTAINING THE D.O.F. NUMBERS OF THE NODES

C I JDIAG(I) = VECTOR CONTAINING THE ADDRESS OF THE DIAGONAL TERMS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IN THE GLOBAL STIFFNESS MATRIX 'SKG'

WSK = NUMBER OF TERMS IN THE GLOBAL STIFFNESS MATRIX

MB = HALF BANDWIDTH OF THE STIFFNESS MATRIX

IOUT = OUTPUT DEVICE NUMBER

COMMON BLOCKS

MDP(X, J) = MEMBER INCIDENCES

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

CALL ELINFO (ELNUM, ITYPE, NODE, IFLAGS, ISTART, LINES)

MINDOF = 10000000

DO 120 NODE = 1, IREL
        ID = MDP*Z(NODE, ELMUM) - 1
        ID1 = ID + DI
        ID2 = ID + DF(ID)*ID
        IF (ID2 .LT. 0) MINDOF = MIN(MINDOF, ID2)

110 CONTINUE

120 CONTINUE

AT THIS POINT THE ID OF EACH COLUMN IS STORED IN JDIAG

DO 110 ID1 = 1, IREL*MB
        ID2 = ID1*DF
        JDIAG(ID2) = MINO(JDIAG(ID2), MINDOF)

110 CONTINUE

110 CONTINUE

110 CONTINUE
modify for mpc locate smallest def number

IF (MPMC .GT. 0) THEN
  DO 170 K1 = 1, MPMC
  I1 = MDOF + K1
  ILOC = MPCIADR(I1,K1)
  DO 160 K2 = ILOC, ILOC + MPCIADR(2,K1) - 1
    IF (MPCIADR(K2,K1) .GT. 0) THEN
      ID = I1DOF(MPCIADR(K2))
      IF (ID .GT. 0) JDIAK(I1) = MISO(JDIAK(I1),ID)
    ENDIF
  DO 170 K1 = 1, MPMC
  I1 = MDOF + K1
  IF (ID .GT. 0) JDIAK(I1) = ID - JDIAK(K1) + 1
ENDIF

CONTINUE

DO 180 K2 = 1, MDOF
  IF (IFLAG2 .EQ. 0) THEN
    NMT = 1
    ID = 0
    DO 190 K = 1, NTDF + 1
      NMT = NMT + 1
      ID = JDIAK(K) + NMT
    DO 180 K2 = 1, MDOF
      NMT = JDIAK(K2) + NMT
    ELSE
      ID = 0
      DO 200 K = 1, NTDF
        ID = ID + JDIAK(K)
        JDIAK(K) = ID
      ENDIF
      NTSK = 2*JDIAK(NTDF)
  ELSE
    NTSK = JDIAK(NTDF+1) - JDIAK(1)
  ENDIF

WRITE (LOUT, 900) MDOF, NTDF, KTSK
RETURN
900 FORMAT(//1X,'NUMBER OF dofs = ', 12, 1X,'active dofs ', 12, 1X,'SIZE OF THE STIFFNESS MATRIX ','=', 12)
END

***************
INCLUDE(PROCESS)
SUBROUTINE UNTSK(INTOPS,ELNUM,WEL,THICKL,ZSI,LTHICK,ZS)
IMPLICIT REAL*8(A-H,O-Z)
...SWITCHES: RESNUM=100:10,FORMAT=900:10
...SWITCHES:
***************
SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
***************
NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
***************
INTEGER ELNUM
REAL*8 B,NXT,BETA,HSI,SI,LTHICK

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C (PROGRAM:
SUBROUTINE CHTRL(SKG, SKGL,R,IDOF, JDIAG, HTSK, HTDF, TOUT, mband)
C ARGUMENT LIST:
CI SKG(I) = GLOBAL STIFFNESS MATRIX STORED AS A ONE
CI R(I) = LOAD VECTOR
CI IDOF(I) = VECTOR CONTAINING THE D.O.F. NUMBERS OF JOINTS
CI JDIAG(I) = LOCATION OF THE DIAGONAL TERMS OF EACH COLUMN
CI HTSK = TOTAL NUMBER OF TERMS IN THE 'SKG' MATRIX
CI HTDF = NUMBER OF TOTAL D.O.F. IN THE PROBLEM
CI TOUT = OUTPUT DEVICE
CI MBBAND = HALF BAND WIDTH OF THE STIFFNESS MATRIX
C
THICKL = 0.0
ZSI = 0.0
DO 100 K1 = 1, HTSK
    THICKL = THICKL + N(K1, IHTGPH) * LTHICK(K1)
    IF (HIPXI .EQ. 1) THEN
        ZSI = ZSI + N(K1, IHTGPH) * ZS(K1)
    ELSE IF (HIPXI .EQ. 2) THEN
        IP = IHTGPH / (HIPXI * HIPETA)
        ZSI = ZSI + N(K1, IHTGPH) * (ZS(K1) - LTHICK(K1) * 0.57735 / 2.0)
    ELSE IF (HIPXI .EQ. 3) THEN
        ZSI = ZSI + N(K1, IHTGPH) * ZS(K1)
    ELSE IF (HIPXI .EQ. 4) THEN
        ZSI = ZSI + N(K1, IHTGPH) * (ZS(K1) + LTHICK(K1) * 0.57735 / 2.0)
    EHDIF
    ELSE IF (HIPXI .EQ. 1) THEN
        ZSI = ZSI + N(K1, IHTGPH) * (ZS(K1) - LTHICK(K1) * 0.7745966 / 2.0)
    ELSE IF (HIPXI .EQ. 2) THEN
        ZSI = ZSI + N(K1, IHTGPH) * (ZS(K1) + LTHICK(K1) * 0.7745966 / 2.0)
    ELSE IF (HIPXI .EQ. 3) THEN
        ZSI = ZSI + N(K1, IHTGPH) * ZS(K1)
    ELSE IF (HIPXI .EQ. 4) THEN
        ZSI = ZSI + N(K1, IHTGPH) * (ZS(K1) + LTHICK(K1) * 0.7745966 / 2.0)
    EHDIF
100 CONTINUE
RETURN
END
C ============================= CONT 1 =============================
C INCLUDE (PROCESS)
SUBROUTINE CHTRL(SKG, SKGL, R, IDOF, JDIAG, HTSK, HTDF, TOUT, mband)
COMMON BLOCKS

REFFER TO THE COMMON BLOCK DESCRIPTIONS.

========================================================================

IMPLICIT REAL*(8(A-H,O-Z)

SWITCHES: RENUM=100:10,FORMAT=800:10

SWITCHES:

SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

RECOV RESTAT RCOVST DIAGEL DISCOS ASSEMB
ANGEL SOLVE2 SOLVE1 GETSTR SWAP3 CHREC
ICOPY STORE OUTPUT ARCHIV

========================================================================

LOGICAL YES,NO
COMMON/TRANS/DC(3,3)
COMMON/INPUT/IPSP(10000)
COMMON/MA11(100000),RE(60000)
COMMON/MA12/UTIL(60000)
COMMON/MA14/RE(60000)
COMMON/INPUT7/B1T(60000),UMSC(60000),UMB(60000)
COMMON/INPUT/BODES,HELM,MBDF,EL1B1,MB1T,IFLAG1,IFLAG2,IDIM,1
110000000000000000000000000000000000000000000000000000000000000
COMMON/INCB11/FRACT(10),EL1BC(10),LDCST,ISCPT
COMMON/INPUT2/FC1,FA11,FA12,FA13,FA14,EDIV1,ISTOP
COMMON/INPUT3/IFLAG3,ID1T,IFPLOT
COMMON/CUST1/ISCREM,BIT
COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LKEEP,LDEV5
COMMON/LILGTM/IFLAG5
COMMON/SAFE4/I9DF(60000),IUDDF(60000)
COMMON/PATH/BISTX(10,50),BISTY(10,50),IPATH(10
COMMON/TEMPS/TEMPS(100000),TEMPS(60000)
COMMON/MPC0S/CDEFMP(40000),ALML(60000),MPCDFG(40000),
$MPC0R(2,5000),EMPC,MPCST,MAXPC

DIMENSION R(1),SKG(1),SKGL(1),IDOF(1),IDIA(1),DUMMY(3)

BTIME = 0

BP = 0
FRACST = 0
ICHECK = 0
YES = .TRUE.
NO = .FALSE.
ZAP = BRDF/IDIM

MDF = MAXIMUM DEGREES OF FREEDOM INCLUDING THE SUPPORTS

MDF = BODES+MBDF

IF (ISTR = 0) IOSTRA = BL1BC(ISCPT)

IF THIS RUN IS A RESTART THEN RESTORE THE LAST CONVERGED VALUES
OF THE EQUILIBRIUM LOAD VECTOR AND THE TOTAL DISPLACEMENT VECTOR

IF (IFLAG3 .EQ. 1) THEN
CALL RECOV (FRACST, MDF, I9START, BTDF, IDOF)
BPREV = FRACST
CALL RESTAT
CALL ROCHST
CALL DIAGSL (SELEM, BDF, IDDF, IDOF, JDIAG, BTK, Bnodes,
1 IFLAG2, IOUT)
IFINAL = ISTART + BLINC(IHCPtr)
ISTART = ISTART + 1
ISAVE = IFLAG1
ELSE
ISTART = 1
IFINAL = BLINC(IHCPtr)
ENDIF
FOR THE FIRST ITERATION OF THE FIRST INCREMENT USE THE
GEOMETRIC LINEARITY ROUTINES.
IFLAG1 = 0; GEOMETRIC LINEARITY
= 1; GEOMETRIC NON-LINEARITY
ISAVE = DUMMY VARIABLE USED TO STORE THE VALUE OF 'IFLAG1'
ISAVE = IFLAG1
IFLAG1 = 0
ENDIF
CALCULATE THE PROPER LOAD OR DISPLACEMENT INCREMENT
UIINC( K ) = APPLIED INCREMENT OF DISPLACEMENT
U( K ) = TOTAL APPLIED DISPLACEMENTS
R( K ) = TOTAL APPLIED LOADS
RINC( K ) = INCREMENT OF APPLIED LOADS
RE( K ) = EQUILIBRIUM LOAD VECTOR
BLINC = NUMBER OF LOAD INCREMENTS
ICOUNT = ITERATION COUNT FOR THE RUN
IOCNT = INCREMENT COUNT FROM THE THE START OR SINCE THE LAST
OUTPUT. WHEN 'IOCNT' IS EQUAL TO 'IOINTR' A COMPLETE
OUTPUT WILL BE GENERATED.
DO IM = 1, MDF
TEMP1(IM) = U(IM)
TEMP2(IM) = R(IM)
END DO
110 CONTINUE
IF (IHCPtr .GT. 1) THEN
ISTART = IFINAL + 1
IFINAL = BLINC(IHCPtr) + ISTART - 1
ENDIF
IOCNT = 0
IPLCNT = 0
DO 340 INCREM = ISTART, IFINAL
IOCNT = IOCNT + 1
IPLCNT = IPLCNT + 1
RFACT = 1.000
RFSUM = 0.000
FAC = FACLOW
FACFAC = FACHIGH
FAC = CONVERGENCE FACTOR
FACLOW = LOWEST ALLOWABLE CONVERGENCE FACTOR
FACHIGH = LARGEST ALLOWABLE CONVERGENCE FACTOR

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C FACESW = NEW CONVERGENCE FACTOR CALCULATED BY ROUTINE `CHECK`
C RFACT = REDUCTION FACTOR FOR SUBINCREMENTATION
C U(K) = INCREMENT OF THE APPLIED DISPLACEMENTS USED
C FOR THE FIRST ITERATION
C RIT(K) = TOTAL APPLIED LOAD AT THE END OF THE INCREMENT
C
C*VDIR: PREFER VECTOR
DO 160 K1 = 1, MDF
ATIME = 0.0
BTIME = 0.0
IF (ISCPR .EQ. 1) THEN
ATIME = FRACT(ISCPR)
BTIME = ATIME*INCREN/IFINAL
IF (IFLAGS .EQ. 1) THEN
ATIME = FRACT(ISCPR) - FRACST
BTIME = FRACST + ATIME*(INCREN-ISTART)+1)/HSTC(1)
ELSE IF (ISCPR .GT. 1) THEN
ATIME = FRACT(ISCPR) - FRACT(ISCPR-1)
BTIME = FRACT(ISCPR-1) + ATIME*(INCREN-ISTART+1)/HSTC(1)
ENDIF
160 CONTINUE
ENDIF
IHFB = 0
IJHFH = 0
HUH1 = 0
HUH2 = 0
FRACT1 = 0.0
FRACT2 = 0.0
IF (IRDOF(K1).NE.0) IHFB = IRDOF(K1)
IF (IJUFO(K1).NE.0) IJHFH = IJUFO(K1)
IF (IRHFB .NE. 0) HUH1 = IPATH(IRFBH)
IF (IJHFB .NE. 0) HUH2 = IPATH(IJHFB)
IF (HUH1 .GT. 1) THEN
DO IP1 = 1, HUH1 - 1
IF (BTIME .GE. HISTX(IRFBH,IP1) .AND. BTIME .LE. HISTX(IRFBH,IP1)) THEN
FRACT1 = (BTIME*(HISTX(IRFBH,IP1)-HISTX(IRFBH,IP1+1)) + (HISTX(IRFBH,IP1)*HISTX(IRFBH,IP1+1))) / (HISTX(IRFBH,IP1)-HISTX(IRFBH,IP1+1))
GO TO 130
ENDIF
END DO
130 CONTINUE
ENDIF
160 CONTINUE
IF (NUM2 .GT. 1) THEN
DO IP2 = 1, NUM2 - 1
IF (BTIME .GE. HISTX(IJUFBH,IP2) .AND. BTIME .LE. HISTX(IJUFBH,IP2)) THEN
FRACT2 = (BTIME*(HISTX(IJUFBH,IP2)-HISTX(IJUFBH,IP2+1)) + (HISTX(IJUFBH,IP2)*HISTX(IJUFBH,IP2+1))) / (HISTX(IJUFBH,IP2)-HISTX(IJUFBH,IP2+1))
GO TO 150
ENDIF
150 CONTINUE
UC(K1) = RFACT*TEMP1(K1)*FRACT2
RIT(K1) = RFACT*TEMP2(K1)*FRACT1
UHIC(K1) = UC(K1)
RICH(K1) = RIT(K1)
DO 320 K1 = 1, NBODES
   I = HEDF*(K1-1)
   DO 210 ILOOP = 1, TREP
       ILOOPT = ILOOP - 1
       ICODE = IABS(ISPB(K1), ILOOP)
       I = I + IDIM*ILOOPT
   CONTINUE

DO 220 K2 = 1, IDIM
   IDIR = I + K2
   DUMMY(K2) = RIT(IDCIR)
170 CONTINUE

IF (ICODE .GT. 0) THEN
   CALL Dircos (ICODE, IDIM, K1)
C*VDIR: PREFER SCALAR
C
DO 180 K3 = 1, IDIM
   CST = 0.
   CST = CST + RIT(IDIR)*DC(K3,K2)
180 CONTINUE
   DUMMY(K2) = CST

DO 190 K2 = 1, IDIM
   CST = 0.
   CST = R(IDIR)*DC(K3,K2)
   DUMMY(K2) = CST
190 CONTINUE

IF (ID .GT. 0) R(ID) = DUMMY(K2)
200 CONTINUE

IF (NHPC .GT. 0) THEN
   DO IMPC = 1, NHPC
      ALAMB(IHPC) = R((NTDF-NMPC)+IMPC)
   ENDIF
C
CALL ASSEMB(ESK, SGL, R, U, IDOF, JDIA, ETDF, MBAND, IOUT)
C
CALL RENV
C
IF (IFLAG2 .EQ. 0) THEN
   CALL SOLVE2 (SKG, R, JDIA, ETDF, 1, IOUT)
   CALL SOLVEV (SKG, R, JDIA, ETDF, 2, IOUT)
   ELSE IF (IFLAG2) THEN
      CALL SOLVEV (SKG, SGL, R, JDIA, ETDF, YES, YES)
   ENDIF
C
IF (NHPC .GT. 0) THEN
   DO IMPC = 1, NHPC
      ALAMB(IMPC) = R((ETDF-NHPC)+IMPC)
ESD DO
ESDIF
C
C*VDIR: PREFER SCALAR
C
DO 240 K1 = 1, MDF
   ID = IDOF(K1)
   IF (ID .GT. 0) U(K1) = U(K1) + R(ID)
240    CONTINUE
C
DO 300 K1 = 1, NODES
   I = NDF*(K1-1)
   DO 290 ILOOP = 1, IREP
       ICODE = IAND(IRESP(K1),ILOOP)
       I = I + IDIM*ILOOP
C*VDIR: PREFER SCALAR
   DO 260 K2 = 1, IDIM
       IDIR = I + K2
       DUMMY(K2) = U(IDIR)
   260    CONTINUE
   IF (ICODE .GT. 0) THEN
       CALL DIRCOS (ICODE, IDIM, K1)
C*VDIR: PREFER SCALAR
   DO 280 K2 = 1, IDIM
       IDIR = I + K2
       UTOTAL(IDIR) = DUMMY(K2)
   280    CONTINUE
   CONTINUE
   300    CONTINUE
C
IFLAGl = ISAVE
C*VDIR: PREFER VECTOR
DO 310 K1 = 1, MDF
   RE1(K1) = RE(K1)
   U(K1) = 0.
   U(RE(K1)) = U(WC(K1))
   RE = RE(K1)
310    CONTINUE
C
CALL GETST (IOUT)
CALL RESS
C
C
check the convergence of the procedure for each element of
reinforced concrete.
C
ICHECK = 0
CALL SWAP3
CALL CHKRC (ISCREM, HIT, IERROR, IOUT, ICHECK)
IF (ICHECK .EQ. 0) THEN
  CALL ILOCOPY
  CALL REWIN
  GO TO 330
ENDIF

320 CONTINUE

END OF ITERATION LOOP

330 CONTINUE
IF (ICHECK .EQ. 0 .OR. HIT .GE. HHIT) THEN
  WRITE (IOUT, 910) IHCREM
  IF (HIT .GT. HHIT .AND. ICHECK .EQ. 1) THEN
    CALL REWIN
    WRITE (IOUT, 920) HIT
    CALL STORE (SKG, IOUT, IERROR)
    CALL REWIN
    CALL ARCHIV (MDF)
    STOP
  EHDIF
  BPREV = BTIHE
  CALL STORE (BTIHE, HDF, IHCREM, HTDF, IDDF)
  WRITE (IOUT, 920) HIT
  CALL OUTPUT (SKG, IOUT, IERROR)
  CALL REWIN
  EHDIF
  340 CONTINUE
  ICNTPTR = ICNTPTR + 1
  IDINTR = 0
  IFLAG3 = 0
  IF (ICNTPTR .LE. LDCOHT) GO TO 110
  CALL ARCHIV (MDF)
  RETURN

C C
C
900 FORMAT(35X, 'NO CONVERGENCE ')
910 FORMAT(35X, 'INCREMENT NO.', I7, ')
920 FORMAT(35X, 'ITERATION NO.', I7, ')
C1002 FORMAT(/1X, 'TOTAL NUMBER OF ITERATIONS FOR THIS RUN IS ')
  C 1 , = ', I6)
930 FORMAT(/1X, '------ OUTPUTS ARE FOR THE LAST CONVERGED INCREMENT ', I4)
940 FORMAT(/1X, '------- OUTPUTS AT INCREMENT ', I4)
950 FORMAT(/1X, '------ ALLOWABLE NUMBER OF ITERATIONS EXCEEDED AT ',
  1 ' LOAD STEP ', I4/9X, 'THE EFFECTIVE CONVERGENCE FACTOR ', E10.3,
  2 ' IS WITHIN TOLERANCE '/9X, 'EXECUTION CONTINUES ')
960 FORMAT(/1X, '------ ALLOWABLE NUMBER OF DIVerging ITERATIONS ',
  1 ' IS EXCEEDED AT LOAD STEP ', I4/9X,
  2 ' EXCEEDS THE PRISCRIBED TOLERANCE '/9X, 'SUBINCREMENT HAS FAILED TO CORRECT THE PROBLEM'/
  3 ' EFFECTIVE CONVERGENCE FACTOR EXCEEDS THE PRISCRIBED TOLERANCE '/9X, 'SUBINCREMENT HAS FAILED TO CORRECT THE PROBLEM'/
  4 ' SUBINCREMENT HAS FAILED TO CORRECT THE PROBLEM'/
970 FORMAT(/1X, '------ ALLOWABLE NUMBER OF DIVERGING ITERATIONS ',
  1 ' EXCEEDED AT LOAD STEP ', I4/9X,
  2 ' EXCEEDS THE PRISCRIBED TOLERANCE '/9X, 'SUBINCREMENT HAS FAILED TO CORRECT THE PROBLEM'/
  3 ' EFFECTIVE CONVERGENCE FACTOR EXCEEDS THE PRISCRIBED TOLERANCE '/9X, 'SUBINCREMENT HAS FAILED TO CORRECT THE PROBLEM'/
  4 ' SUBINCREMENT HAS FAILED TO CORRECT THE PROBLEM'/
980 FORMAT(/1X, '------ ALLOWABLE NUMBER OF DIVERGING ITERATIONS ',
  1 ' EXCEEDED AT LOAD STEP ', I4/9X,
990 FORMAT(/I1, '***** ALLOWABLE NUMBER OF ITERATIONS EXCEEDED AT', 
1 ' LOAD STEP ', 16/9X, 'THE EFFECTIVE CONVERGENCE FACTOR', E10.3, 
2 ' EXCEEDS THE PRISCRIBED TOLERANCE'/9X, 'REDUCTION FACTOR IS', 
3 'REDUCES TO ' ,PE.6/9X, 'EXECUTION CONTINUES')
END

C INCLUDE(PROCESS)
SUBROUTINE TSHRCELNUM.NNEL, INTGPN, THICKE, ITYPE)
IMPLICIT REAL'S (A-H,O-Z)
C . . . SWITCHES : REHUMB=100;10.FORMAT=900:10
C . . .SWITCHES:
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C GTLTK  DMATCS
C
INTEGER ELMUN
REAL=8 LTHICK
REAL=4 THICKE
COMMON/INPUTS/GIPXI, BiPETA, BiPSI, BIP, ITPICOD
COMMON/LAYERB/LTHICK(9),2S(9),DCS(3,3),NLAYRS,MATRL,LYNUM
COMMON/BLOCKS/BISTO(27,70,15),CRACK(27,250,15),IBISTI,IBISTA
$.IH IST2
COMMON/INPUT/THICK(9),IVALG
COMMON/TSHKL/I1(6000,9),FI(6000,9)
COMMON/MATERI/DEP(6,6)
DIMENSION A(9),B(9),C(9),D(9)

II = NIPXI*BIPETA
IPOINT = INTGPN/II
INDEX = INTGPN - IPOINT*II
IF (INDEX .EQ. 0) INDEX = II

IF (LYNUM.EQ.1 .AND. INTGPN.LE.II) THEN
A(INDEX) = 0.0
B(INDEX) = 0.0
C(INDEX) = 0.0
D(INDEX) = 0.0
E1(ELNUM,INDEX) = 0.0
F1(ELNUM,INDEX) = 0.0
ENDIF

ZSI = 0.0
THICKL = 0.0
CALL GTLTK (I1TGP, ELMUN, BECL, TBICKE, ZSI, LTHICK, ZS)
I0P = 1
IOUT = 13
ICODE = 0
CALL DMATCS (ELNUM, ITYPE, INTGPN, IFLAG, IOUT, ICODE, I0P)

ZDIST = THICKL/2.0 + ZSI
A(INDEX) = THICKL*ZDIST*DEP(1,1) + A(INDEX)
B(INDEX) = THICKL*DEP(1,1) + B(INDEX)
C(INDEX) = THICKL*ZDIST*DEP(2,2) + C(INDEX)
D(INDEX) = THICKL*DEP(2,2) + D(INDEX)

IF (LYNUM .EQ. NLAYRS) THEN
ISIG = 1
ELSE
268

IS IG = 0
ED D IF
C
IF

( I S I G .E Q . 1 ) TEES
E l ( E L B U M ,I H D E X ) = A ( I B D E X ) / B ( I H D E X )
F1 (E LD U M ,I BD EX ) = C ( IH D E X ) / D ( I D D E X )
ED D IF
RETURH
EBD
C
C ===================== B L O C K
C
C

D A T A

= == = = = = = = = = = = = = = = = = = = = = = = =

IHCLUDE (PROCESS)
BLOCK DATA
I M P L I C I T R E A L 'S ( A - H , 0 - Z )
RE A L« 4 X I I , ET A I , S I I , FMAG, DMAG, P X I , PETA
CHARACTER'SO GTITL E
CHARACTER*40 S C F P , O F P , O F H , VOLSER

C
C
C

COHSTAHT VALUES ASSIGHED TO CERTAIB VARIABLE DAMES.

1

CDMM0H/IHPUT8/DB0DES, BE L EM ,B B DF ,B LI D Cl , MBIT, I F L A G l , I F L A G 2 , I D I M ,
DIBOD E, BCOLOR,BFREE
COMMON/IBCRl 1 /F RA CT ( 1 0 ) .D L I H C ( I O ) , LDCOHT, IHCPTR
COMMOH/IHPUTB/FAC, FACHEV, FACLOW, FACHIG, EH RG l, HDIVER, IS TD P
C0M M 0 D /ID P UT G/ IF L AG 3, l O I H T R , IFPLOT
C 0 M M 0 H /U T I L 3 /B R E C ( 3 ) ,DWMAX
COMMOB / F I L E l / K E E P , I D E L , S C F P , OFP , OFH, VOLSER
COMMOB/ADMATl/AD ( 8 1 )
COMMOB/DEVICE/LDEVl ,LDEV 2.LDEV3,LDEV4,LDEVB,LDKEEP,LDEV,LDEVST
C O H M O D /E L L IB 1 /X II(2 0 ),E T A I(2 0 ),S II(2 0 )
C 0 M M 0 H / E L L I B 2 / P X K 9 ) ,P E T A ( 9 )
C0MM0B/HERM/H(4 , 4 ) , G X ( 4 , 6 ) , G F ( 4 , 6 )
C 0M M 0D /G R A P H 1/IS (62),IE (62)
COMMOH/GRAPHE/FHAG, DMAG, BI GHT , AHGLE. BO LIDE, I T H I C K , HLIDES , SLID
COMMOB/GRAPH7/GTITLE( 2 0 )
C0HM0B/GRAPH8/LDEVP
C 0M M 0H /E X T R P1/IB T33(9),IB T22(4)
C 0M M 0H /C A ED S1/I204(4), 1 2 0 8 ( 8 ) , 1 3 0 8 ( 8 ) , 1 3 2 0 ( 2 0 )

C
DATA S C F P , O F P , D F D , V O L S E R / ' ' . '
>/
DATA K E E P , I D E L / 0 , 0 /
DATA ( ( H ( I , J ) , J = l , 4 ) , I = l , 4 ) / 2 . , ' 2 . , 1 . , 1 . , - 3 . , 3 . , - 2 . , - l . , 0 . . 0 . ,
1
1 .,0 .,1 .,0 .,0 ,,0 ,/
DATA L D E V B / 1 6 / , L D E V P / 1 7 /
DATA L D E V l , LDEV2, LDEV3, LDEV4, LDEVST/1 , 2 , 3 , 4 , 1 4 /
DATA ( X I I ( K ) , K = 1 , 2 0 ) / - 1 . , 1 . , 1 . , - 1 . , - 1 . , 1 . , 1 . , - l . , 0 . , 1 . , 0 . , - 1 . ,
1
-1 .,1 .,1 .,-1 .,0 .,1 .,0 .,-1 ./
DATA ( E T A I ( K ) , K = 1 , 2 0 ) / - 1 . , - 1 . , 1 . , 1 . , - 1 . , - l . , l . , 1 . , - 1 . , 0 . , 1 . ,
1
0 .,- 1 .,- 1 .,1 .,1 .,- 1 .,0 .,1 . ,0 ./
DATA ( D L I B C ( K ) , K = 1 , 1 0 ) / 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 /
DATA ( F R A C T ( K ) , K = 1 , 1 0 ) / 1 . , 1 . , 1 . , 1 . . 1 . , 1 . , 1 . , 1 . , 1 . , 1 . /
DATA ( S I I ( K ) , K = 1 , 2 0 ) / - 1 . , - 1 . , - 1 . , - 1 . , 1 . , 1 . , 1 . , 1 . , - 1 . , - 1 .
1
0 .,0 .,0 .,0 .,1 .,1 .,1 . ,1 ./
DATA HHODES, BELEM,BBDF,MBIT, I F L A G l , I F L A G 2 , I D I M / 0 , 0 , 2 , 1 , 0 ,
1
0 , 2 / , IFLAG3, lO IB TR , IF PL O T ,B FR EE/0, 0 , 0 , 0 /
DATA B D I V E R , F A C / 1 , . 0 0 1 / , A D / 8 1 ' 0 . /
DATA H R E C / l , l , l / , B W M A X / 0 /
DATA P X I / - 1 . , 1 . , 1 . , - 1 . , 0 . , 1 . , 0 . , - 1 . , 0 . /
DATA P E T A / - 1 . , - 1 . , 1 . , 1 . , - 1 . , 0 . . 1 . , 0 . , 0 . /
C
C
C

GRAPHICS ELEMEHT L I B E COHECTIVITY DATA

Reproduced with perm ission of the copyright owner. Further reproduction prohibited without permission.


DATA IS/1,2,3,4,1,5,2,3,4,
  1       1,6,2,6,3,7,4,8,1,2,3,4,5,6,7,8,2,3,4,
  2       1,9,2,10,3,11,4,12,5,17,6,18,7,19,8,20,1,15,4,16,3,15,2,14,
  3       1,8,6,3,7,3,8,4,9/
DATA IS/2,3,4,1,6,2,3,4,1,
  1       6,2,6,3,7,4,8,1,2,3,4,1,6,7,8,5,6,7,8,6,
  2       9,2,10,3,11,4,12,1,17,6,18,7,19,8,20,6,13,5,16,8,16,7,14,6,
  3       5,8,2,7,3,8,4,9,1/
DATA FMAG,DMAG,HIGH,ANGLE,ESKLINE,ITYICK/1.,1.,0.08,.0,.0/.2/
DATA ELINES,MLE/0,0/
C
C GAUSS point to node connectivity data for nodal extrapolation
C
DATA IDT33/1,3,9,7,2,6,8,4,5/,INT22/1,2,4,3/
C
C ELEMENT CONNECTIVITY DATA FOR CAEDS ELEMENTS
C
DATA ID204/1,2,3,4/, ID208/1,5,2,6,3,7,4,8/, ID308/1,2,3,4,5,6,7,8/, 
  DATA ID204/1,9,2,10,3,11,4,12,13,14,15,1,17,6,18,7,19,8,20/, 
  END
C
C shouldBe CAEDS element data
C
C SUBROUTINE CAEDS(IDOF,U,R,lOUT, UHIVER)
C
C PROGRAM:
C
CAEDS is used to read the Universal Files which have been
created by the CAEDS Graphics Finite Element Modeler module.
C
C ENTRY:
C
C     IDOF = array containing the degree of freedom
C     IDOF = is the array containing the degree of freedom
C
C     U      = is the array containing the degree of freedom
C     IDOF = is the array containing the degree of freedom
C     IDOF = is the array containing the degree of freedom
C     IDOF = is the array containing the degree of freedom
C     IDOF = is the array containing the degree of freedom
C
C RETURN:
C
C IMPLICIT REAL*8 (A-E,O-Z)
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C REAL*8, B, H, U, W
C
C Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
REAL*4 XYZ, IAX, ORIG, IZPLB
CHARACTER*80 BUFFER, ID1, ID2, ID3, ID4
CHARACTER*40 UDIVER, UHV, RSHAHE, CSNAME
COMMON/INPUT2/WIP(20,6000)
COMMON/INPUTS/XYZ(3,10000)
COMMON/INPUTS/UW}(10),NUZ(10),UX(10),UY(10),EZ(10),
1 PIX(10),PIY(10),PIZ(10),PX(10),PY(10),PZ(10)
COMMON/OUTPUT/WITX(10), WGTY(10), WITZ(10)
COMMON/OUTPUT/IFLAG, IF1CAED
COMMON/CAEDS2/ID1, ID2, ID3, ID4, LOADCM

C ---- Common INPUT2 is used with a different organization in module
C SAFE. It is used here for the purpose of saving storage for
C holding some temporary parameters. Array dummy is used to
C adjust the size of this common block to match the size specified
C in module SAFE.

C COMMON/INPUT2/IDEF(10000), IDISP(10000), ORIG(3,10000), IAX(3,10000),
1 IZPLB(3,10000), ICTYPE(10000), IREF(10000),
2 SPYHES(10000), DUMMY(10000)

C ---- IDEF = definition coordinate system number for the node
C IDISP = displacement coordinate system number
C ORIG = coordinates of the origin
C IAX = coordinates of a point on the x-axis
C IZPLB = coordinates of a point on the xz-plane
C ICTYPE = coordinate type
C IREF = reference coordinate system of the current coord. sys.
C SPYHES = physical property number of the element
C
COMMON/INPUT8/BRDES, HELEM, NELEM, BLIBC, MNTIT, IFLAG1, IFLAG2, IDIM,
1 BIBODE, COLOR, FREEE
COMMON/ICR0/FRAC(10), BLIBC(10), LCOST, ICRPTR
COMMON/INPUT1/INFUEL(6,5000)
COMMON/INPUTS/ISPB(10000)
COMMON/INPUTS/ATYPE(10)
COMMON/INPUTS/MTYPE(10)
COMMON/INPUTS/MTYPE(10)
COMMON/MFACFR(10000)
COMMON/MCPC/coefmp(40000), ALAMB(40000), MPCNDF(40000),
1 MPCDR(2,5000), nmpc, mpccnt, maxmpc
DIMENSION IDDF(99), U(99), R(99), SVAR(8)
DIMENSION IDB(8), ISV(6)
EQUIVALENT(8, RSHAHE, CSNAME)

C ---- IF1CAED is the flag that indicates to the output module
C that the requested outputs should also be provided in the
C form of a CAEDS universal file.
C
NUM = 0
ITYPE = 0
IBTCOD = 0
NIPSE = 0
NIPETA = 0
NIPX1 = 0
LIBES = 0
ISTARY = 0
ITYPE1 = 0
IFLAG = 0
IF1CAED = 1
ID2 = 'NLRC3D STATIC ANALYSIS'

C ---- Open the CAEDS universal file
C
UNV = '/\'//UNIVER
OPEN(19, STATUS='UNKNOWN', FILE='UBV')

HEIGH = 0
HLow = 100000
HIGHT = 0
LLow = 100000

IOFLAG = IBSET(IOFLAG,20)
IOFLAG = IBSET(IOFLAG,22)

IDSAT = 0
CONTINUE

ICOUNT = 0
IF (IDSAT .EQ. 0) THEN
READ (19, '(A80)', END=300) BUFFER
IF (BUFFER(5:6) .EQ. ('-1')) IDSAT = 1
GO TO 100
ELSE IF (IDSAT .EQ. 1) THEN
READ (19, *, END=310) ITYPE
C
IF (ITYPE .EQ. 1) THEN
GO TO 120
ELSE IF (ITYPE .EQ. 18) THEN
GO TO 130
ELSE IF (ITYPE .EQ. 71) THEN
GO TO 140
ELSE IF (ITYPE .EQ. 749) THEN
GO TO 290
ELSE IF (ITYPE .EQ. 752) THEN
GO TO 180
ELSE IF (ITYPE .EQ. 754) THEN
GO TO 180
ELSE IF (ITYPE .EQ. 756) THEN
GO TO 270
ELSE
ICOUNT = 0
CONTINUE
READ (19, '(A80)', END=310) BUFFER
ICOUNT = ICOUNT + 1
IF (BUFFER(5:6) .EQ. ('-1')) GO TO 110
C
ICOUNT = ICOUNT - 1
WRITE (IOUT, 900) ICOUNT, ITYPE
IDSAT = 0
GO TO 100
ENDIF

C --- Data set 15: NODES
C --- Dataset Organization
C
Record 1:   FORMAT(4(I10,1P3E13.5)
Field 1    -- Node label
Field 2    -- Definition coord. system number
Field 3    -- Displacement coord. system number
Field 4    -- color
Fields 5-7  -- 3-Dimensional coordinate of the
             -- node in the definition system

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Record 1 is repeated for each node in the model.

CONTINUE
READ (19, '(a80)', END=310) BUFFER
IF (BUFFER(5:6) .NE. ('=')) THEN
   ICOUNT = ICOUNT + 1
C
--- ISKIP = dummy variable to read the color of the node
C
READ (BUFFER, '(4I10,1P,3E13.5)') NODE, IDEF(NODE), IDISP(NODE)
   1 , ISKIP, XYZ(1,NODE), XYZ(2,NODE), XYZ(3,NODE)
C
--- Set bit 10 of ISP to 1 to indicate that the position of the
C
node has been defined.
C
ISP(NODE) = IBSET(ISP(NODE),10)
NEIGH = MAX(NEIGH,NODE)
SLow = MIN(SLOW,NODE)
GO TO 120
ELSE
   WRITE (IOUT, 910) ICOUHT, ITYPE, NEIGH, SLOW
   IDSAT = 0
   GO TO 100
ENDIF
C
--- Data set 18: Coordinate Systems
C
--- Dataset Organization
C
Record 1: FORMAT(810)
Field 1 -- Coordinate system number
Field 2 -- Coordinate system type
   =0 - cartesian
   =1 - cylindrical
   =2 - spherical
Field 3 -- Reference coordinate system num.
Field 4 -- Color
Field 5 -- Method of definition
   =1 - origin, +X axis, +YZ plane
C
Record 2: FORMAT(2012)
Field 1 -- Coordinate system name
C
Record 3: FORMAT(1P6E13.5)
Total of nine coordinate system definition param.
Field 1-3 -- Origin of new system in reference system
Field 4-6 -- Point on +X axis of the new system specified in reference system
Field 5 -- Point on +Z plane of new system specified in reference system
C
Record 1 through 3 are repeated for each coordinate system in
the model.
C
CONTINUE
READ (19, '(a80)', END=310) BUFFER
IF (BUFFER(5:6) .HE. ('='-1')) THEN
ICOUNT = ICOUNT + 1
READ (BUFFER,✳(I=10)) NUM, ICTYPE(NUM), IREF(NUM), ISKIP,
METRO
READ (19,✳) BUFFER
READ (19,✳(I=13)) (ORIG(I,NUM), I = 1, 3), (XAX(I,NUM)
1)
READ (19,✳(I=13)) (XZPLS(I,NUM), I = 1, 3)
GO TO 130
ELSE
IDSAT = 0
WRITE (IOUT, 920) ICOUHT, ICTYPE
GO TO 100
ENDIF

--- Data set 71: Elements

Table of Element Names and Type Numbers

<table>
<thead>
<tr>
<th>Elem_Type</th>
<th>N_Nodes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>2</td>
<td>Rod</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>Linear Beam</td>
</tr>
<tr>
<td>22</td>
<td>2</td>
<td>Tapered Beam</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>Curved Beam</td>
</tr>
<tr>
<td>24</td>
<td>3</td>
<td>Parabolic Beam</td>
</tr>
<tr>
<td>31</td>
<td>2</td>
<td>Straight Pipe</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>Curved Pipe</td>
</tr>
<tr>
<td>41</td>
<td>3</td>
<td>Plane stress Triangle</td>
</tr>
<tr>
<td>42</td>
<td>6</td>
<td>Plane stress parabolic triangle</td>
</tr>
<tr>
<td>43</td>
<td>9</td>
<td>Plane stress cubic triangle</td>
</tr>
<tr>
<td>44</td>
<td>4</td>
<td>Plane stress linear quadrilateral</td>
</tr>
<tr>
<td>45</td>
<td>8</td>
<td>Plane stress parabolic quadrilateral</td>
</tr>
<tr>
<td>46</td>
<td>12</td>
<td>Plane stress cubic quadrilateral</td>
</tr>
<tr>
<td>51</td>
<td>3</td>
<td>Plane strain Triangle</td>
</tr>
<tr>
<td>52</td>
<td>6</td>
<td>Plane strain parabolic triangle</td>
</tr>
<tr>
<td>53</td>
<td>9</td>
<td>Plane strain cubic triangle</td>
</tr>
<tr>
<td>54</td>
<td>4</td>
<td>Plane strain linear quadrilateral</td>
</tr>
<tr>
<td>55</td>
<td>8</td>
<td>Plane strain parabolic quadrilateral</td>
</tr>
<tr>
<td>56</td>
<td>12</td>
<td>Plane strain cubic quadrilateral</td>
</tr>
<tr>
<td>61</td>
<td>3</td>
<td>Flat plate linear triangle</td>
</tr>
<tr>
<td>62</td>
<td>6</td>
<td>Flat plate parabolic triangle</td>
</tr>
<tr>
<td>63</td>
<td>9</td>
<td>Flat plate cubic triangle</td>
</tr>
<tr>
<td>64</td>
<td>4</td>
<td>Flat plate linear quadrilateral</td>
</tr>
<tr>
<td>65</td>
<td>9</td>
<td>Flat plate parabolic quadrilateral</td>
</tr>
<tr>
<td>66</td>
<td>12</td>
<td>Flat plate cubic quadrilateral</td>
</tr>
<tr>
<td>71</td>
<td>4</td>
<td>Membrane linear quadrilateral</td>
</tr>
<tr>
<td>72</td>
<td>6</td>
<td>Membrane parabolic triangle</td>
</tr>
<tr>
<td>73</td>
<td>9</td>
<td>Membrane cubic triangle</td>
</tr>
<tr>
<td>74</td>
<td>3</td>
<td>Membrane linear triangle</td>
</tr>
<tr>
<td>75</td>
<td>8</td>
<td>Membrane parabolic quadrilateral</td>
</tr>
<tr>
<td>76</td>
<td>12</td>
<td>Membrane cubic quadrilateral</td>
</tr>
<tr>
<td>81</td>
<td>3</td>
<td>Axisymmetric linear triangle</td>
</tr>
<tr>
<td>82</td>
<td>6</td>
<td>Axisymmetric parabolic triangle</td>
</tr>
<tr>
<td>84</td>
<td>4</td>
<td>Axisymmetric linear quadrilateral</td>
</tr>
<tr>
<td>85</td>
<td>6</td>
<td>Axisymmetric parabolic quadrilateral</td>
</tr>
<tr>
<td>91</td>
<td>3</td>
<td>Thin shell linear triangle</td>
</tr>
<tr>
<td>92</td>
<td>6</td>
<td>Thin shell parabolic triangle</td>
</tr>
<tr>
<td>93</td>
<td>9</td>
<td>Thin shell cubic triangle</td>
</tr>
<tr>
<td>94</td>
<td>4</td>
<td>Thin shell linear quadrilateral</td>
</tr>
<tr>
<td>95</td>
<td>8</td>
<td>Thin shell parabolic quadrilateral</td>
</tr>
</tbody>
</table>
Thick shell cubic quadrilateral
Thin shell linear wedge
Thin shell parabolic wedge
Thin shell cubic wedge
Thin shell linear brick
Thin shell parabolic brick
Solid linear tetrahedron
Solid linear wedge
Solid parabolic wedge
Solid cubic wedge
Solid linear brick
Solid parabolic brick
Solid cubic brick
Solid parabolic wedge
Solid linear brick
Solid parabolic brick
Solid cubic brick
Solid parabolic tetrahedron
Rigid bar
Rigid element
Node to node translational spring
Node to node rotational spring
Node to ground translational spring
Node to ground rotational spring
Node to node damper
Node to ground damper
Node to node gap
Node to ground damper
Lumped mass
Axisymmetric linear shell
Axisymmetric parabolic shell
Constraint element

### Dataset Organization

Record 1: 
```
FORMAT(110)
Field 1 -- Element label
Field 2 -- FE graphical description id
Field 3 -- FE type id
Field 4 -- Physical property table number
Field 5 -- Material property table number
Field 6 -- Color
Field 7 -- Number of nodes on element
```

Record 2: 
```
FORMAT(610)
Field 1-n -- Node labels defining element
```

Record 1 and 2 are repeated for each element in the model.
READ (19, '(8110)') (HOP(I,HUM), I = 1, 8)
ELSE IF (IDTYPE.EQ.45.OR.IDTYPE.EQ.65.OR.IDTYPE.EQ.85) THEN
    ITYPE1 = 208
    ISTART = 10
    LINES = 8
    NIPXI = 3
    NIPETA = 3
    NIPS1 = 0
    IHTCOD = 0
    READ (19, '(8110)') HOP(I,HUM), HOP(5,HUM), HOP(2,HUM),
     1  HOP(6,HUM), HOP(3,HUM), HOP(7,HUM), HOP(4,HUM), HOP(8,
     2  NUM)
ELSE IF (IDTYPE .EQ. 115) THEN
    ITYPE1 = 308
    ISTART = 18
    LINES = 12
    NIPXI = 2
    NIPETA = 2
    NIPS1 = 2
    IHTCOD = 0
    READ (19, '(8110)') (HOP(I,HUM), I = 1, 8)
ELSE IF (IDTYPE .EQ. 116) THEN
    ITYPE1 = 320
    ISTART = 30
    LINES = 24
    NIPXI = 3
    NIPETA = 3
    NIPS1 = 3
    IHTCOD = 0
    READ (19, '(8110,2/(8110))') HOP(1,HUM), HOP(9,HUM), HOP(2,
     1  ,NUM), HOP(10,HUM), HOP(3,HUM), HOP(11,HUM), HOP(4,HUM)
     2  , HOP(12,HUM), HOP(13,HUM), HOP(14,HUM), HOP(16,HUM)
     3  , HOP(16,HUM), HOP(5,HUM), HOP(17,HUM), HOP(6,HUM),
     4  HOP(18,HUM), HOP(7,HUM), HOP(19,HUM), HOP(6,HUM), HOP(20,
     5  ,NUM)
ELSE IF (IDTYPE .EQ. 94) THEN
    ITYPE1 = 304
    ISTART = 1
    LINES = 4
    NIPXI = 2
    NIPETA = 2
    NIPS1 = 2
    IHTCOD = 0
    READ (19, '(8110)') HOP(1,HUM), HOP(2,HUM), HOP(3,HUM),
     1  HOP(4,HUM)
ELSE IF (IDTYPE .EQ. 95) THEN
    ITYPE1 = 308
    ISTART = 10
    LINES = 8
    NIPXI = 2
    NIPETA = 2
    NIPS1 = 2
    IHTCOD = 0
    READ (19, '(8110)') HOP(1,HUM), HOP(5,HUM), HOP(2,HUM),
     1  HOP(6,HUM), HOP(3,HUM), HOP(7,HUM), HOP(4,HUM), HOP(8,
     2  NUM)
ELSE
    CALL ERRORS (9, 1, 'CAEDS ')
ENDIF
C
C    Set bit 20 of ISPB to 1 to indicate that the element connect.
C    has been defined
C
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ISP8(NUM) = IBSET(ISPB(NUM),20)

IF (IDTYPE.GE.40 .AND. IDTYPE.LT.90) THEN
  IFLAG = 1
ELSE IF (IDTYPE.GE.50 .AND. IDTYPE.LT.60) THEN
  IFLAG = 2
ELSE IF (IDTYPE.GE.80 .AND. IDTYPE.LT.90) THEN
  IFLAG = 3
ELSE IF (IDTYPE.GE.90 .AND. IDTYPE.LT.100) THEN
  IFLAG = 4
ENDIF

IFL = HHEL+8 + ITYPE1*S12 + IFLAG*262144 + 2097152*HIPZI
     16777216*HIPETA + 134217728*NIPSI + MAT

INF2 = ISTART + 131072*LINES + INTCOD

INFOEL(1,NUM) = INF1
INFOEL(2,BUM) = INF2

LHIGH = MAXO(LHIGH,HUH)
LLOW  = MINO(LLOW,NUH)
GO TO 140
ELSE
  WRITE (lOUT, 930) ICDUNT, ITYPE, LHIGH, LLLOW
  IDSAT = 0
  GO TO 100
ENDIF

---- Dataset 762: Permanent groups

---- Dataset Organization

Record 1:   FORMAT(6I10)
  Field 1  -- group number
  Field 2  -- active constraint set number
  Field 3  -- active restraint set number
  Field 4  -- active load set number
  Field 5  -- active DOF set number
  Field 0  -- number of entities in group

Record 2:   FORMAT(20A2)
  Field 1  -- Group name

Record 3-8:  FORMAT(81I10)
  Field 1  -- entity type code
  Field 2  -- entity tag
  Field 3  -- entity type code
  Field 4  -- entity tag
  Field 5  -- entity type code
  Field 6  -- entity tag
  Field 7  -- entity type code
  Field 8  -- entity tag

Records 3 and 4 are repeated for each nodal force of the loadset

ENDO CONTINUE

READ (19, '(A80)', EED=310) BUFFER
IF (BUFFER(5:6) .NE. ("-1")) THEN
  READ (BUFFER, '(6I10)') IGNUP, I1, I2, I3, I4, NUMBER
  NFREE = MAXO(NFREE,IGGROUP)
  READ (19, '(A40)') RSAME
IF (MOD(NUMBER,4) .EQ. 0) THEN
  K2 = NUMBER/4
ELSE
  K2 = NUMBER/4 + 1
ENDIF

DO K1 = 1, K2
  READ (19, '(8I10)') (HVAR(K3), K3 = 1, 8)
  DO ID = 1, 7, 2
    IF (HVAR(ID) .EQ. 0) THEN
      ITAG = HVAR(ID+1)
      IFBODY(ITAG) = IBSET(IFBODY(ITAG),IGROUP)
    ENDIF
  END DO
END DO
WRITE (lOUT, 980) NUMBER, ITYPE, IGROUP, RSHAME
GO TO 180
ELSE
  IDSAT = 0
  GO TO 100
ENDIF

C ---- DATASET 754: CONSTRAINT SETS
C
C ---- DATASET ORGANIZATION
C
RECORD 1:  FORMAT(2I10)
  FIELD 1 -- CONSTRAINT SET NUMBER
  FIELD 2 -- CONSTRAINT TYPE
  FIELD 3 =0 -- EMPTY SET
  =1 -- COUPLED DOFS
  =2 -- MULTI-POINT CONSTRAINTS

RECORD 2:  FORMAT(20A2)
  FIELD 1 -- CONSTRAINT SET NAME

FOR CONSTRAINT TYPE 1 - COUPLED DOFS

RECORD 3:  FORMAT(3I10,6I12)
  FIELD 1 -- INDEPENDENT NODE LABEL
  FIELD 2 -- COLOR NUMBER
  FIELD 3 -- NUMBER OF DEPENDENT NODES
  FIELD 4-9 -- SWITCHES FOR 1-6
    =0 -- OFF
    =1 -- ON

RECORD 4+H:  FORMAT(8I10)
  FIELD 1-8 -- DEPENDENT NODE LABELS

RECORD 3 AND 4 ARE REPEATED FOR EACH NODE IN THE CONSTRAINT SET.

FOR CONSTRAINT TYPE 2 - MULTI-POINT CONSTRAINTS

RECORD 3:  FORMAT(4I10,1P,2E13.8,I10)
  FIELD 1 -- EQUATION LABEL
  FIELD 2 -- NUMBER OF TERMS
  FIELD 3 -- FORCE/DISPLACEMENT SWITCH
c
  = 0 - FORCE
  = 1 - DISPLACEMENTS

  FIELD 4 -- COLOR
  FIELD 5 -- REAL PART OF FORCE/DISP. CONSTANT
  FIELD 6 -- IMAGINARY PART OF FOR./DISP. CONS.
  FIELD 7 -- DATA TYPE
     = 0 - REAL
     = 1 - COMPLEX

  RECORD 4+3: FORMAT((1I0,12,1I2,3E13.6))
  FIELD 1 -- NODE LABELS
  FIELD 2 -- NODEL DOFS.
    0 -- SCALAR
    1 -- X DISP.
    2 -- Y DISP.
    3 -- Z DISP.
    4 -- X ROT.
    5 -- Y ROT.
    6 -- Z ROT.

  FIELD 3 -- REAL PART OF CONSTRAINT COEFFS.
  FIELD 4 -- NODE LABELS
  RECORDS 3 AND 4+8 ARE REPEATED FOR EACH CONSTRAINT SET.
  THIS INCLUDES SEPARATORS, AND THE DATA SET TYPE RECORDS FOR
  EACH TYPE OF EACH SET.

180 CONTINUE
  READ (19, '(21I0)') HUB, IDTYPE
  READ (19, '(A40)') CSNAME

190 CONTINUE
  READ (19, '(A40)', END=310) BUFFER
  IF (BUFFER(6:6) .NE. '-1') THEN
    IF (IDTYPE .EQ. 1) THEN
      READ (BUFFER, '(3110, 612)') IND1, ICOLOR, HDN, (ISW(I), I = 1, 6)
      IDENT1 = (IND1-1)*NNDF
      DO 220 K1 = 1, NREC
        READ (19, '(8110)') (IDN(I), I = 1, 8)
      DO 210 K2 = 1, 8
        IF (IDN(K2) .GT. 0) THEN
          IDENT2 = (IDN(K2)-1)*NNDF
          DO 200 K3 = 1, NHDF
            IF (ISW(K3) .EQ. 1) THEN
              COEFL = 1.
              IF (KDC(NHDF) .NE. 0) THEN
                NHDF = NHDF + 1
              ENDIF
              D2 = (IDN(K2)) * (ISW(I), I = 1, 6)
              D2 = (IDN(K2)) * (ISW(I), I = 1, 6)
              IF (IDN(K2) .GT. 0) THEN
                IDENT2 = (IDN(K2)-1)*NHDF
                DO 200 K3 = 1, 6
                  IF (ISW(K3) .EQ. 1) THEN
                    ICOUNT = ICOUNT + 1
                    E1MPC = E1MPC + 1
                    MPC1P = MPC1P + 1
                    MPCADD(1, E1MPC) = MPC1P
                    MPCADD(2, E1MPC) = 2
                    MPCADD(MPC1P) = IDENT1 + K3
                    COEFL = MPC1P + 1
                    MPC1P = MPC1P + 1
                    MPCADD(MPC1P) = IDENT2 + K3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
COEFMP(MPCPHT) = -1.0
ENDIF
CONTINUE

ENDIF
CONTINUE
ELSE IF (IDTYPE .EQ. 2) THEN
  READ (BUFFER, '(4I10,I1P,2E13.6,I10)') LEQ, BTERMS, ICASE, 1
  ICOLOR, CSTR, CSTI, IXDATA
C
  LEQ EQUATION LABELS
  BTERMS NUMBER OF TERMS
  ICASE FORCE/DISP. SWITCH. 1 FORCE 2 DISPLACEMENTS
  CSTR CONST. PART OF EQUATION.
  IXDATA DATA TYPE 0 REAL 1 COMPLEX
C
  ICOUHT = ICOUHT + 1
  MPCM = MPCM + 1
  MPCPNT = MPCPHT + 1
  MPCADR(HMPC) = 0
  CDEFMP(MPCPHT) = -CSTR
  MPCADRCl,HMPC) = MPCPHT
  MPCADR(2,HHPC) = HTERM S + 1
C
  DO 230 K1 = 1, BTERMS
    READ (19, '(I10,1P,2E13.5)') BODE, IDF, CSTR, CSTI
    IF (IDF .LE. HHDF) THEN
      IDEHT1 = (NODE-1)*HHDF + IDF
      MPCPHT = MPCPHT + 1
      MPCSTR = MPCPHT + 1
      MPCDOP(MPCPHT) = IDENT1
      CDEFMP(MPCPHT) = CSTR
    ELSE
      CALL ERRORS (41, 1, 'CAEDS')
    ENDIF
  ENDIF
  CONTINUE
ENDIF
GO TO 190
ELSE
  IDSAT = 0
  WRITE (IDUT, 960) ICOUHT, ITYPE, HUM, CSEAME
  GO TO 100
ENDIF
C
C ----- Data Set 755: Restrains
C
C ----- Data Set Organization
C
C Record 1: FORMAT(2I10)
  Field 1 -- Restrains set number
  Field 2 -- Restrains type
Field 3 =0 - empty set
=1 - nodal displacements
=2 - nodal temperature

C Record 2: FORMAT(20A2)
  Field 1 -- Restrains set name
For restraint type 1 - nodal displacements

C Record 3: FORMAT(2I10,7I2)
  Field 1 -- Restrains label
Field 2 -- Color number
Field 3-8 -- Switches for physical DOFs 1-6
Field 9 -- Switch for user defined DOF

Record 4: FORMAT (1P6, E13.8)
Field 1-6 -- Displacement DOFs 1-6

Record 3 and 4 are repeated for each node in the restraint set.

240 CONTINUE
READ (19, '(2110)') NUM, IDTYPE
READ (19, '(A40)') RSHAME
IF (IDTYPE .EQ. 2) CALL ERRORS (10, 1, 'CAEDS ')

250 CONTINUE
READ (19, '(A80)', EHD=310) BUFFER
IF (BUFFER(B:6) .HE, ( ' - ! ' ) )  THES
ICOUHT = ICOUHT + 1
READ (BUFFER, '(2110,712)') NODE, ICOLOR, (IDOF(I,BODE), I = 1
1 , BSDF)

Identify the node as a support
ISPB(HODE) = IBSET(ISPB(H0DE),18)
READ (19, '(1P,6E13.5)') (U(I,HDEDE), I = 1, HHDF)
DO 1 = 1, HHDF
IF (CU(I,HQDE) .BE. 0.0 )  IDDF(I,HGDE) = -IDOF(I,HODE)
END DO
GO TO 250
ELSE
IDSAT = 0
WRITE (lOUT, 970) ICOUHT, ITYPE, HU, RSHAME
GO TO 100
ENDIF

Dataset 756: Load sets

Dataset Organization

Record 1: FORMAT(2110)
Field 1 -- Load set number
Field 2 -- Load type
=0 -- empty set
=1 -- nodal force
=2 -- nodal temperature
=3 -- finite element face pressure
=4 -- finite element face heat flux
=5 -- finite element edge pressure
=6 -- finite element edge heat flux
=7 -- nodal heat source
=8 -- finite element edge convect.
=9 -- finite element edge radiation
=10 -- finite element face convect.
=11 -- finite element face radiation
=12 -- finite element dist. heat generation
Record 2: FORMAT(20A2)
Field 1 -- Load set name

For load type 1 - nodal forces

Record 3: FORMAT(2I10,6I2)
Field 1 -- nodal force label
Field 2 -- Color number
Field 3-8 -- Switches for DOFs 1-6
  =0 - off
  =1 - on

Record 4: FORMAT(1P,6E13.6)
Field 1-6 -- Forces for DOFs 1-6

Records 3 and 4 are repeated for each nodal force of the loadset

270 CONTINUE
READ (19, '(2I10)') LOADCH, IDTYPE
READ (19, '(40)') RSNAM
IF (IDTYPE .NE. 1) CALL ERRORS (17, 1, 'CAEDS ')

280 CONTINUE
READ (19, '(40)', ESD=310) BUFFER
IF (BUFFER(5:6) .EQ. ' -!') THEN
  ICOUNT = ICOUNT + 1
  READ (BUFFER, '(2I10,2I2)') NODE, ICOLOR
  READ (19, '(1P,6E13.6)') (R(I,HDDE), I = 1, HHDF)
  GO TO 280
ELSE
  IDSAT = 0
  WRITE (IOUT, 960) ICOUNT, ITYPE, HUM, RSNAM
  GO TO 100
ENDIF

290 CONTINUE
READ (19, '(2I10)') MODEL, HDRK
READ (19, '(40)') I01
READ (19, '(40)', ESD=310) BUFFER
IF (BUFFER(5:6) .EQ. ' -!') THEN
  IDSAT = 0
  WRITE (IOUT, 940) ITYPE, MODEL, I01
  GO TO 100
ENDIF

---- Close the CAEDS universal file

300 CONTINUE
CLOSE(UNIT=19)
NELEM = LEIG
NRODES = RHIG
RETURN

310 CONTINUE
CALL ERRORS (0, 1, 'CAEDS ')

900 FORMAT(/IX,'-------- A total of ',I7,' records in ',
  1 ' dataset ',I3,' were skipped. ')
910 FORMAT(/IX,'-------- A total of ',I6,' nodes in dataset ',I3,
  1 ' were processed. '/
  2 9X,'Highest Node Label = ',I6/)
282

3 9X,'Lowest Node Label =',16
920 FORMAT(/1X,'------- a total of ',16,' coordinate systems',
1    ' in dataset ',13,', were processed.')
930 FORMAT(/1X,'------- A total of ',16,' elements in dataset ',
1    ' were processed.')
940 FORMAT(/1X,'------- A total of ',16,' records for',
1    ' dataset ',13,', were processed.')
950 FORMAT(/1X,'------- A total of ',16,' elements in',
1    ' dataset ',13,', were processed.')

960 FORMAT(/1X,'--------- A total of ',16,' records for',
1    ' dataset ',13,', were processed.')
970 FORMAT(/1X,'--------- A total of ',16,' restraint nodes for',
1    ' dataset ',13,', were processed.')
980 FORMAT(/1X,'--------- A total of ',16,' node loads for',
1    ' dataset ',13,', were processed.')

END

C ------------------------------------------- OUT20 -------------------------------------------
C
C INCLUDE (PROCESS)
C
SUBROUTINE OUT20(IDUT,ICASE)
C
PROGRAM:
C
OUT20 outputs various stresses and strains at the nodes on
C elements in the CASES universal file format.
C
ARGUMENT LIST:
C
IDUT = output device number of the ELRIC output file
C ICASE = 1: output second Piola-Kirchhoff stresses
C         = 2: output Cauchy stresses
C         = 3: output total strains
C         = 4: output elastic strains
C         = 5: output plastic strains
C
IMPLICIT REAL*8 (A-H,O-Z)
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C REVIS ELISFO ELISM LMISFO LAGRO IGET
C
C*******************************************************************************
CHARACTER*1 CTEMP
CHARACTER*80 ID1, ID2, ID3, ID4
INTEGER ELNUM
REAL*4 THICK
REAL*8 LTHICK
COMMON/UTIL/VALUE(25),CTEMP
COMMON/INPUT1/SIPXI, HIPETA, HIPSI, HIP, IINTOD
COMMON/INPUT8/NDODES, NELM, NDIF, HHEL, ELHUM, IFLAG1, IFLAG2, IDIM,
1 HINODE, HIPCOLOR, HPFREE
COMMON/INPUT11/PRACT(10), HINOD(10), LCCT, IECPTK
COMMON/INPUT/MATYPE(10)
COMMON/ISPC01/P(64, 27)
COMMON/DEVICE/LDEV1, LDEV2, LDEV3, LDEV4, LDEV5, LDEV6, LDEV7, LDEV8
COMMON/CASEDS1/1040, 1208(8), 1308(8), 1320(20)
COMMON/CASEDS2/1ID1, ID2, ID3, ID4, LOADCH
COMMON/LAYERS/LTHICK(9), ZS(9), DCS(3, 3), NLAYRS, MATR, LYHUB
C
C ---- Identification headers
C
C ID1 = FE_model name
C ID2 = run identification
C ID3 = run date and time
C ID4 = description of the data
C
DIMENSION VALS(6, 20), HFD(20), MAXLYR(18)
DATA ZERO/0.0000/
C
C ---- Identify the output request
C
ISTART = 0
IEED = 0
IVAL = 0
IF (ICASE .EQ. 1) THEN
   ID4 = 'SECOND PIOLA KIRCHHOFF STRESSES'
   ISTART = 0
   IVAL = 2
ELSIF (ICASE .EQ. 3) THEN
   ID4 = 'TOTAL STRAINS'
   ISTART = 6
   IVAL = 3
ELSIF (ICASE .EQ. 4) THEN
   ID4 = 'ELASTIC STRAINS'
   ISTART = 12
   IVAL = 3
ENDIF
C
DO IM = 1, 16
   MAXLYR(IM) = 0
ENDDO
CALL REVIS
DO 130 ELNUM = 1, NELM
   CALL ELISFO (ELNUM, ITYPE, NELM, IFLAG, IST, LINES)
   CALL ELISM (ELNUM, IDEHT, IINTOD, HIPX1, HIPETA, HIPSI,
1 MATNUM, THICK)
C
DO 120 LYNUM = 1, NLAYRS
   CALL LYISFO (LYNUM, LTHICK, ZS, MATR, DCS, NELM, ELNUM,
1 HIPXI, HIPETA, HIPSI)
C ---- Write the universal data set start code -1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IF (MAXLYR(LYNUM) .EQ. 1) GO TO 110
MAXLYR(LYNUM) = 1
WRITE (20 + LYNUM, '(A6)') ' -1'

C Write the dataset identification code 67

WRITE (20 + LYNUM, '(A6)') ' 57'

C Write the five ID LINES. (empty at the moment)

C ID1 = model identification
C ID2 = run identification
C ID3 = run date/time
C ID4 = load case name
C ID5 = (lH Load case number; I10)

WRITE (20 + LYNUM, 900) ID1, ID2, ID3, ID4, LOADCH
FORMAT(6,I10)

C Write record G: FORMAT(6,I10)

C Field 1: model type 0: unknown
1: structural
2: heat transfer
3: fluid flow

C Field 2: analysis type 0: unknown
1: static
2: normal mode
3: complex eigenvalue
4: transient
5: frequency response
6: buckling

C Field 3: data characteristic 0: unknown
2: 3 DOF global translation vec.
3: 6 DOF global trans. and rot. v.
4: symmetric global tensor
5: general global tensor
6: shell and plate element stress resultant

C Field 4: specific data type (IV4L)
0: unknown
1: unknown
2: stress
3: strain
4: element force
5: temperature
6: heat flux
7: strain energy
8: displacement
9: reaction force
10: kinematic energy
11: velocity
12: acceleration
13: strain energy density
14: kinematic energy density
15: hydrostatic pressure
16: heat gradient
17: code checking value
18: coefficient of pressure
19: ply stress
20: ply strain
Field 5: data type
1: integer
2: single precision floating point
3: double precision floating point
4: single precision complex
5: double precision complex

Field 6: number of data values for each position on the element

WRITE (20 + LYBML, 910) IVAL

Record 7 for analysis type = 1, static FORMAT(8I10)
Field 1: 1
Field 2: 1
Field 3: Load case number
Field 2: 1

WRITE (20 + LYBML, 920)

Record 8 for analysis type = 1, static format (6E13.5)
Field 1: 0.0
WRITE (20 + LYBML, '(E13.5)') ZERO

Records 9:

Field 1: Element number (ELBML)
Field 2: Data expansion code (IEXP)
1: data present for all nodes
2: data present for 1st node, all other nodes are the same
Field 3: Number of nodes on element (NBML)
Field 4: Number of data values per node (NVPN)
Record 10: FORMAT(6E13.5)

Fields 1-n: data values at node 1 (NVPN real or complex valu.)

For IEXP = 1 record 10 is repeated NBML times
For IEXP = 2 record 10 is repeated once.

Records 9 and 10 are repeated for all elements

CONTINUE
CONTINUE
CONTINUE
IEXP = 3
NVPN = 6
DO 280 ELBML = 1, NELEM
   CALL ELINFO (ELBML, ITYPE, NBML, IFLAG, IST, LIBES)
   CALL ELINFOH (ELBML, IDEHT, IHTCOD, NIPX, HIPETA, HIPS, HTHUS, 1)
   MATHUS, THICK)

DO 260 LYHUM = 1, HLAYRS
   CALL LVINFO (LYHUM, LTHICK, 25, MATRL, DCS, NEEL, ELBMIN, 
   KIPXI, KIPXII, LTHICK)
   IF (ITYPE .LE. 0) GO TO 270
   IF (ITYPE .LT. 0 .OR. IDENT .LT. 0) THEN
      CALL LLAGGI (ITYPE, IFLAG, NEEL, IERROR)
      IF (ITYPE .GT. 300) THEN
         IF (IFLAG .EQ. 0) THEN
            DO 140 K1 = 1, NEEL
            HID(K1) = I320(K1)
            140 CONTINUE
         ELSE IF (ITYPE .EQ. 308) THEN
            DO 150 K1 = 1, NEEL
            HID(K1) = I308(K1)
            150 CONTINUE
         ENDIF
         ELSE IF (ITYPE .EQ. 308) THEN
            DO 160 K1 = 1, NEEL
            HID(K1) = I204(K1)
            160 CONTINUE
         ELSE IF (ITYPE .EQ. 308) THEN
            DO 170 K1 = 1, NEEL
            HID(K1) = I208(K1)
            170 CONTINUE
         ENDIF
         ELSE IF (ITYPE .GT. 200) THEN
            IF (ITYPE .EQ. 204) THEN
               DO 180 K1 = 1, NEEL
               HID(K1) = I204(K1)
               180 CONTINUE
            ELSE IF (ITYPE .EQ. 208) THEN
               DO 190 K1 = 1, NEEL
               HID(K1) = I208(K1)
               190 CONTINUE
            ENDIF
            ELSE IF (ITYPE .LE. 0) THEN
               DO 200 K2 = 1, IEND
               VALS(K2, K1) = 0.
               200 CONTINUE
            ELSE IF (ITYPE .LE. 0) THEN
               DO 210 K1 = 1, NEEL
               DO 220 K2 = 1, IEND
               VALS(K2, K1) = 0.
               210 CONTINUE
               DO 240 ISTGPE = 1, IEIP
               CALL IGGET (LDEV1, 96, '(496)', 6)
               DO 230 K1 = 1, NEEL
               COXST = P(ISTGPE, K1)
               DO 220 K2 = 1, IEND
               IF (LYHUM .LE. 1) WRITE(20 + LYHUM, '(4T10)') ELBMIN, IEIP, NEEL, NVP
               220 CONTINUE
               240 CONTINUE
            ENDIF
         ENDIF
      ENDIF
   ENDIF
C
   WRITE (20 + LYHUM, '(4T10)') ELBMIN, IEIP, NEEL, NVP

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C ---- Process expansion code
C
DO 260 I = 1, NSEL
   ID = NID(I)
   IF (ITYPE .GT. 300) THEN
      WRITE (20 + LYHUM, '(G13.5)') VALS(1, ID), VALS(4, ID), VALS(2, ID), VALS(6, ID), VALS(8, ID)
   ELSE IF (ITYPE .GT. 200) THEN
      WRITE (20 + LYHUM, '(G13.5)') VALS(1, ID), VALS(3, ID), VALS(2, ID), ZERO, ZERO, VALS(4, ID)
   ENDIF
260 CONTINUE
C
C ---- Close the universal dataset
C
DO IM = 1, 16
   MAXLYR(IM) = 0
END DO
DO 320 ELNUM = 1, NSEL
   CALL ELIHFO (ELHUM, ITYPE, HHEL, IFLAG, 1ST, LINES)
   CALL ELIHTM (ELHUM, IDEHT, IHTCOD, HIPXI, HIPETA, HIPSI, MATHUM, THICK)
310 CONTINUE
C
C DO 310 LHYUM = 1, LAYRS
   CALL LYIHFO (LYHUM, LTHICK, ZS, MATRL, DCS, NSEL, ELNUM, HIPXI, HIPETA, HIPSI)
C
C ---- Write the universal dataset start code -1
C
IF (MAXLYR(LYHUM) .EQ. 1) GO TO 300
MAXLYR(LYHUM) = 1
WRITE (20 + LYHUM, '(A80)') ' -1>
300 CONTINUE
310 CONTINUE
320 CONTINUE
CALL REWIH
C
C =========================== D  U  T 2 1 ==============================
C
C INCLUDE (PROCESS)
SUBROUTINE OUT21OUT
IMPLICIT REAL*8 (A-H,O-Z)
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C # NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C*************************************************************************
C CHARACTER*80 ID1,ID2,ID3,ID4
COMMON/INPUTS/RENUM,NELEM,HEDF,HLINC,NMRT,I1FLG1,I1FLG2,IDI1,
WRITE (20, '(A6)') ' -1 '

WRITE (20, '(A6)') ' 55'

ID1 = model identification
ID2 = run identification
ID3 = run date/time
ID4 = description of results
ID5 = (17H Load case number; 110)

IVAL = 0
ID4 = 'BODAL DISPLACEMENTS'
WRITE (20, 900) ID1, ID2, ID3, ID4, LOADCH

Field 1: model type
0: unknown
1: structural
2: heat transfer
3: fluid flow

Field 2: analysis type
0: unknown
1: static
2: normal mode
3: complex eigenvalue
-3: complex eigenvalue
4: transient
5: frequency response
6: buckling
7: complex eigenvalue \( \times \) second order

Field 3: data characteristic
0: unknown
1: scalar
2: 3 DOF global translation vec.
3: 6 DOF global trans. and rot. v.
4: symmetric global tensor
5: general global tensor
6: shell and plate element stress resultant

Field 4: specific data type (IVAL)
0: unknown
1: general
2: stress
3: strain
4: element force
5: temperature
6: heat flux
7: strain energy
Field 5: data type
1: integer
2: single precision floating point
4: double precision floating point
5: single precision complex
6: double precision complex

Field 6: number of data values for each position on the element

WRITE (20, 910) IVAL

Record 7 for analysis type =1, static FORMAT(8I10)

Field 1: 1
Field 2: 1
Field 3: Load case number

WRITE (20, 920)

Record 8 for analysis type =1, static format (6E13.6)

Field 1: 0.0

WRITE (20, '(E13.6)') ZERO

Records 9:
FORMAT(I10)

Field 1: node number (NODE)

Record 10:
FORMAT(6E13.6)

Fields 1-n: data values at node 1 (SDV real or complex valu.)

Records 9 and 10 are repeated for all elements

DO 100 K1 = 1, NNODES
  IF (BTEST(ISPB(K1),10)) THEN
    ID = HHDF*(K1-1)
    WRITE (20, 930) K1, (UTDTAL(ID+K2), K2 = 1, DNDF)
  ENDIF
100 CONTINUE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C "Close the universal dataset"
C
WRITE (20, '(A6)') '-1'

RETURN
900 FORMAT(4(A60,':',LOAD CASE NUMBER,':',I10)
910 FORMAT(9X,'1',9X,'1',9X,'3',9X,'5',9X,'2',9X,'6')
920 FORMAT(9X,'1',9X,'1',9X,'1')
930 FORMAT(I10/6E13.5)
END

C "========================== UT22 ===========================" C
C INCLUDE (PROCESS)
C SUBROUTINE OUT22(IOUT)
IMPLICIT REAL*8 (A-H,O-Z)
C . . .SWITCHES: REM00=100:10,FORMAT=900:10
C . . .SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE C
C IN SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT C
C
C "Character=60 ID1, ID2, ID3, ID4
COMM0/INPUT8/NODES,NELM,WDF,SLINC,NEIT,IFLAG,IFLAG2,IDI4,
1 MNODE,ND00,NDFREE
COMM0/IRCR11/FRACT(5000),SLINC1(150),LDC00ST,ISC0TR
COMM0/INPUT8/ISPCR(100000)
COMM0/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDEV6,LDEV7,
7 LDEV8,LDEV9,LDEV10
COMM0/MAIN4/RE(60000)
COMM0/CAED52/ID1, ID2, ID3, ID4, LOADCN
DIMENSION DUMMY(36)
DATA ZERO/0.000/

C "Write the universal dataset start code -1"
C
WRITE (20, '(A6)') '-1'

C "Write the dataset identification code 55"
C
WRITE (20, '(A6)') '55'

C "Write the five ID LINES. (empty at the moment)"
C
ID1 = model identification
ID2 = run identification
ID3 = run date/time
ID4 = description of results
ID5 = (17H Load case number; I10)

IVAL = 0
ID4 = 'SUPPORT REACTIONS'
WRITE (20, 900) ID1, ID2, ID3, ID4, LOADCN

C "Write record 6: FORMAT(6,I10)"
C
Field 1: model type
0: unknown
1: structural
2: heat transfer
3: fluid flow
Field 2: analysis type
0: unknown
1: static
2: normal mode
3: complex eigenvalue
4: complex eigenvalue
5: transient
6: frequency response
7: buckling
8: complex eigenvalue, second order

Field 3: data characteristic
0: unknown
1: scalar
2: 3 DOF global translation vec.
3: 6 DOF global trans. and rot. v.
4: symmetric global tensor
5: general global tensor
6: shell and plate element stress resultant

Field 4: specific data type (IVAL)
0: unknown
1: general
2: stress
3: strain
4: element force
5: temperature
6: heat flux
7: strain energy
8: displacement
9: reaction force
10: kinematic energy
11: velocity
12: acceleration
13: strain energy density
14: kinematic energy density
15: hydrostatic pressure
16: heat gradient
17: code checking value
18: coefficient of pressure
19: ply stress
20: ply strain
21: failure index for ply
22: failure index for bonding
23: reaction heat flow
24: stress error density
25: stress variation
26: scalar gradient
27: shell and plate element stress resultant

Field 5: data type
1: integer
2: single precision floating point
4: double precision floating point
5: single precision complex
6: double precision complex

Field 6: number of data values for each position on the element
WRITE (20, 910) IVAL
Record 7 for analysis type =1, static FORMAT(8I10)

Field 1: 1
Field 2: 1
Field 3: Load case number

WRITE (20, 920)

Record 8 for analysis type = 1, static format (6E13.6)

Field 1: 0.0

WRITE (20, '(E13.6)') ZERO

Records 9: FORMAT(I10)

Field 1: node number (NODE)

Record 10: FORMAT(6E13.6)

Fields 1:n: data values at node I (NDV real or complex valu.)

Records 9 and 10 are repeated for all elements.

DO 100 K1 = 1, NHODES
IF (BTESTCISPB(K1),18)) THEN
  ID = NHDF*(K1-1)
WRITE (20, 930) K1, (RE(ID+K2), K2 = 1, NHDF)
ENDIF
100 CONTINUE

--- close the universal dataset

WRITE (20, ' (46) ') ' -1>
RETURN

900 FORMAT(4(A80/), 'LOAD CASE NUMBER:', I10)
910 FORMAT(6X,'1', 6X, '1', 6X, '3', 6X, '9X', '12', 6X, '6')
920 FORMAT(6X,'1', 6X, '1', 6X, '1')
930 FORMAT(I10/6E13.5)

C INCLUDE(PROCESS)
SUBROUTINE AESTAT
IMPLICIT REAL*8(A-H, O-Z)

C . . . SWITCHES: REHUB=100:10, FORMAT=900:10
C . . . SWITCHES:
C***************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
REAL*4 THICK
REAL*4 LTHICK
INTEGER ELHUM, ELMH
COMMON/LPROF/PROPER(25,18)
COMMON/INPUTS/NODES, NELEM, NDFF, ELHUM, ELMH, IFLAG1, IFLAG2, IDIM,
$ ELMH, ECOLOR, BFAVE
COMMON/INPUTS/THICK(9), IFLAG
COMMON/INPUTS/THICK(9), IFLAG
COMMON/INPUTS/THICK(9), IFLAG
COMMON/INPUTS/THICK(9), IFLAG
COMMON/LAYER/LTHICK(9), ZS(9), DCS(3,3), NLAYRS, MATRL, LYHUM
COMMON/ABC/TESTF(27,80,15), ITESPG, ITESG1, ITESG2
COMMON/BLOCKS/HISTO(27,70,16), CRACK(27,80,15), IRISTY, IRIST1
$ . IRIST2
COMMON/INCR1/FRACT(10), ELHUMC(10), LCOST, ICPTX

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SUBROUTINE CONTROLS THE RESTART PROCEDURE
AND SETS UP THE APPROPRIATE ARRAYS FOR CONCRETE AND STEEL

C CRACK(L,I,J) = ITH CRACK DATA AT LTH SAMPLE POINT OF JTH LAYER
C FOR EACH ELEMENT
C HISTO(L,I,J) = ITH MATER. HIST. DATA AT LTH SAMPLE POINT OF JTH LAYER
C FOR EACH ELEMENT
C TESTF(L,I,J) = ITH TENS. STIF. DATA AT LTH SAMPLE POINT OF JTH LAYER
C FOR EACH ELEMENT (CONCRETE ONLY)
C IHISTY, IHIST1, IHIST2 ARE FILES USED FOR STOREING ARRAY CRACK
C ITHSPG, ITHSG1, ITHSG2 ARE FILES USED FOR STOREING ARRAY HISTO, TESTF
C
C=====================================================================
C
C IHISTY = 50
C IHIST1 = 60
C IHIST2 = 61
C ITHSPG = 80
C ITHSG1 = 70
C ITHSG2 = 71
C ICP = 62
C
C REWIND IHISTY
C REWIND IHIST1
C REWIND IHIST2
C REWIND ITHSPG
C REWIND ITHSG1
C REWIND ITHSG2
C REWIND ICP
C DO 110 ELHUM = 1, HELEM
C CALL ELINFO (ELHUM, ITYPE, KHEL, IFLAG, ISTART, LINES)
C CALL ELHISTM (ELHUM, IDEHT, INTCODE, HIPXI, HIPETA, HIPSI,
1 MATNUM, THICK)
C DO 100 LYHUM = 1, HLAYRS
C CALL LINFO (LYHUM, LTHICK, ZS, MATRL, DCS, KHEL, ELNUM,
1 HIPXI, HIPETA, HIPSI)
C WRITE( *, *) 'HELEM', HELEM, 'HLAYRS', HLAYRS
C RECOVER ARRAY 'HISTO', 'CRACK', 'TESTF' FOR THE CURRENT ELEMENT.
C THESE ARE REQUIRED FOR THE UPDATING PROCEDURE IN THE CONVERGENCE
C CRITERION EMPLOYED. STORES ALL THE MATERIAL INFO ABOUT POINT.
C
C NIP = NIPXI*HIPETA*HIPSI
C READ (IHIST2) ELNUM, LYNUMB, NGAUS, ((CRACK(L,K,LYHUM),K
1 = 1,250), L = 1, NGAUS)
C IF (ELNUM.NE.ELNUM .OR. LYNUMB.NE.LYNUMB .OR. NGAUS.NE.NIP)
1 THEN
2 WRITE (*, *) 'ERROR READING IHISTY IN RESTAT'
3 WRITE (*, *) 'ELNUM', ELNUM, 'LYNUM', LYNUMB, 'NGAUS', NGAUS
4 STOP
C
C ENDIF
WRITE (IHISTY) ELSUHB, LYSUHB, BGAUS, ((CRACK(L,K,LYBUM),K = 1,260), L = 1, BGAUS)  
1  
WRITE (IHIST1) ELSUHB, LYSUHB, BGAUS, ((CRACK(L,K,LYBUM),K = 1,260), L = 1, BGAUS)  
1  
READ (ITHSG2) ELSUHB, LYSUHB, BGAUS, MATRL, ( (TEHSTF(L,K,LYBUM),K = 1,80), L = 1, BGAUS), ((HISTO(L,K,LYBUM),K = 1,70), L = 1, BGAUS), (PROPER(KM,MATRL), KM = 1, 25)  
1  2  3  
IF (ELSUMB.EE.ELNUM.OR. LYSUB.EE.LYNUM.OR. BGAUS.EE.HIP) THEN  
WRITE (* , *) 'ERROR READING ITHSPG IN RESTAT'  
STOP  
ENDIF  
1  
WRITE (ITHSPG) ELSUHB, LYSUHB, BGAUS, MATRL, ((TEHSTF(L,K,LYBUM),K = 1,80), L = 1, BGAUS), ((HISTO(L,K,LYBUM),K = 1,70), L = 1, BGAUS), (PROPER(KM,MATRL), KM = 1, 25)  
1  2  3  
READ (ICP) ELSUHB, LYSUHB, BGAUS, (AF (ELNUM,LYNUM,KJ), KJ = 1, HIP), (AF2(ELNUM,LYNUM,KP), KP = 1, HIP)  
1  
IF (ELSUMB.EE.ELNUM.OR. LYSUB.EE.LYNUM.OR. BGAUS.EE.HIP) THEN  
WRITE (* , *) 'ERROR READING IC2 IN RESTAT'  
WRITE (* , *) 'ELNUM', ELNUM, 'ELNUM', ELNUM, 'LYNUM', LNUM, 'LYNUM', LNUM, 'HGAUS', HGAUS  
STOP  
ENDIF  
100  CONTINUE  
110  CONTINUE  
RETURN  
END  

C*******************************************************************************  
C INCLUDE(PROCESS)  
C SUBROUTINES CLAR(BREL,ITOPS,ITYPE,BREL,ERROR,TDCS,CL)  
IMPLICIT REAL*8(A-H.O-Z)  
C...SWITCHES: RENUM=100:10,FORMAT=900:10  
C...SWITCHES:  
C*******************************************************************************  
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE  
C*******************************************************************************  
C ISOPOD M2D COORD1 SIMUL  
C*******************************************************************************  
C TO EVALUATE THE CONSISTANT CHARACTERISTIC LENGTH OF A CRACKED...
C SAMPLE POINT IN A CONCRETE LAYER.
C CALLED BY CKHIT
C
REAL*4 THICK,XYZ
REAL*8 N,XI,BETA,BXI
COMMON/INPUTS/THICK(9),IFLAG
COMMON/CONSIS/ROTELC(27,3)
COMMON/INPUTS/BOP(20,5000)
COMMON/TRANS/ DC(3,3)
COMMON/INPUTS/XYZ(3,10000)
DIMENSION FZI(8),BDCS(3,3),XMP(3,8),CX(3),N(20),XI(20),
BETA(20),BXI(20),C(8,8),TEMP(4,8),XMP(3,8)

C CL = 0.0
BEL1 = 0
DO JM = 1, 8
DO JK = 1, 8
C(JM,JK) = 0.0
END DO
TEMP(1,JM) = 0.0
TEMP(2,JM) = 0.0
TEMP(3,JM) = 0.0
TEMP(4,JM) = 0.0
END DO
C COMPUTE THE COORDINATES OF THE CENTER OF THE ELEMENT (LOCAL 0,0,0)
C
IF (ITYPE.GE.300 .AND. IFLAG.EQ.0) THEN
BEL1 = 8
XIC = 0.0
ETAC = 0.0
SIC = 0.0
ITYPE1 = 308
CALL IS0P3D(XIC,ETAC,SIC,N,XI,BETA,BXI,ITYPE1,IERROR)
CX(1) = 0.0
CX(2) = 0.0
CX(3) = 0.0
DO 120 I = 1, 8
NODE = DOP(I,BREL)
CX(1) = CX(1) + N(I)*XYZ(1,NODE)
CX(2) = CX(2) + N(I)*XYZ(2,NODE)
CX(3) = CX(3) + N(I)*XYZ(3,NODE)
120 CONTINUE
C LOCAL BODAL COORDINATES
C
DO 130 II = 1, 8
XMP(1,II) = XYZ(1,BOP(II,BREL)) - CX(1)
XMP(2,II) = XYZ(2,BOP(II,BREL)) - CX(2)
XMP(3,II) = XYZ(3,BOP(II,BREL)) - CX(3)
130 CONTINUE
ELSE IF (ITYPE.GE.300 .AND. IFLAG.EQ.4) THEN
WRITE(*,*) 'SHELL'
BEL1 = 4
XIC = 0.0
ETAC = 0.0
SIC = 0.0
ITYPE1 = 204
CALL N2D(XIC,ETAC,N,ITYPE1,IERROR)
CX(1) = 0.0
CX(2) = 0.0
CX(3) = 0.0
NODE = BOP(1,BREL)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[
\begin{align*}
C_X(1) &= C_X(1) + B(1) \cdot XYZ(1, B0DE) \\
C_X(2) &= C_X(2) + B(1) \cdot XYZ(2, B0DE) \\
C_X(3) &= C_X(3) + B(1) \cdot XYZ(3, B0DE) \\
B0DE &= B0P(2, BREL) \\
C_X(1) &= C_X(1) + B(2) \cdot XYZ(1, B0DE) \\
C_X(2) &= C_X(2) + B(2) \cdot XYZ(2, B0DE) \\
C_X(3) &= C_X(3) + B(2) \cdot XYZ(3, B0DE) \\
B0DE &= B0P(3, BREL) \\
C_X(1) &= C_X(1) + B(3) \cdot XYZ(1, B0DE) \\
C_X(2) &= C_X(2) + B(3) \cdot XYZ(2, B0DE) \\
C_X(3) &= C_X(3) + B(3) \cdot XYZ(3, B0DE) \\
B0DE &= B0P(4, BREL) \\
C_X(1) &= C_X(1) + B(4) \cdot XYZ(1, B0DE) \\
C_X(2) &= C_X(2) + B(4) \cdot XYZ(2, B0DE) \\
C_X(3) &= C_X(3) + B(4) \cdot XYZ(3, B0DE)
\end{align*}
\]

LOCAL NODAL COORDINATES

\[
\begin{align*}
X_{THP\ 1}(1,1) &= XYZ(1, B0P(1, BREL)) - C_X(1) \\
X_{THP\ 1}(1,2) &= XYZ(1, B0P(2, BREL)) - C_X(2) \\
X_{THP\ 1}(1,3) &= XYZ(1, B0P(3, BREL)) - C_X(3) \\
X_{THP\ 1}(2,1) &= XYZ(2, B0P(1, BREL)) - C_X(1) \\
X_{THP\ 1}(2,2) &= XYZ(2, B0P(2, BREL)) - C_X(2) \\
X_{THP\ 1}(2,3) &= XYZ(2, B0P(3, BREL)) - C_X(3) \\
X_{THP\ 1}(3,1) &= XYZ(3, B0P(1, BREL)) - C_X(1) \\
X_{THP\ 1}(3,2) &= XYZ(3, B0P(2, BREL)) - C_X(2) \\
X_{THP\ 1}(3,3) &= XYZ(3, B0P(3, BREL)) - C_X(3)
\end{align*}
\]

DO 150 II = 1, 4
  DO IP = 1, 3
    XTMP(IP, II) = 0.0
    XTMP(IP, II) = XTMP(IP, II) + DC(IP, IP) * XTMP(IP, II)
  END DO
END DO

CONTINUE
ESDIF

C FIND THE PHI VALUES

AL1 = TDCS(1,1)
AM1 = TDCS(1,2)
AF1 = TDCS(1,3)
IF (ITYPE.GE.300 .AND. IFLAG.EQ.4) THEN
  AL1 = DC(1,1) * TDCS(1,1) + DC(2,1) * TDCS(1,2) + DC(3,1) * TDCS(1,3)
  AM1 = DC(1,2) * TDCS(1,1) + DC(2,2) * TDCS(1,2) + DC(3,2) * TDCS(1,3)
  AF1 = DC(1,3) * TDCS(1,1) + DC(2,3) * TDCS(1,2) + DC(3,3) * TDCS(1,3)
END IF
DO 160 I = 1, BREL
  XROT = AL1 * XTMP(1, I) + AM1 * XTMP(2, I) + AF1 * XTMP(3, I)
  PHI(I) = 0.0
  IF (XROT .GE. 0.) PHI(I) = 1.0
END DO
CONTINUE

C FIND THE DERIVATIVES OF THE SHAPE FUNCTIONS (LINEAR ORDER) @ ISTGPN

X1 = 0.0
Y1 = 0.0
Z1 = 0.0
CALL COORDI (BREL, BREL, ISTGPN, ITYPE, X1, Y1, Z1)
IF (ITYPE.GE.300 .AND. IFLAG.EQ.4) THEN
$X_1 = X_1 - CI(1)$
$Y_1 = Y_1 - CI(2)$
$Z_1 = Z_1 - CI(3)$

$X_2 = DC(1,1)X_1 + DC(2,1)Y_1 + DC(3,1)Z_1$
$Y_2 = DC(1,2)X_1 + DC(2,2)Y_1 + DC(3,2)Z_1$
$Z_2 = DC(1,3)X_1 + DC(2,3)Y_1 + DC(3,3)Z_1$

$X_1 = X_2$
$Y_1 = Y_2$
$Z_1 = Z_2$

ERDIF

IF (ITYPE .GE. 300 .AND. IFLAG .EQ. 0) THEN
  BE1 = 8
  DO 170 IM = 1, 8
       C(IM,1) = 1.0
       C(IM,2) = XYZ(1,NDP(IM,BREL))
       C(IM,3) = XYZ(2,NDP(IM,BREL))
       C(IM,4) = XYZ(3,NDP(IM,BREL))
       C(IM,5) = XYZ(1,NDP(IM,BREL)) + XYZ(2,NDP(IM,BREL))
       C(IM,6) = XYZ(1,NDP(IM,BREL)) + XYZ(3,NDP(IM,BREL))
       C(IM,7) = XYZ(1,NDP(IM,BREL)) + XYZ(2,NDP(IM,BREL))
       C(IM,8) = XYZ(1,NDP(IM,BREL)) + XYZ(2,NDP(IM,BREL)) + XYZ(3,
       CONTINUE
C
  EPS = 1.E-20
  DETER = SIMUL(NRCOL,C,EPS,BROW,IOUT)
  IF (DETER .EQ. 0) WRITE (*, *) 'ERROR IN SIMUL SUB-CLEH'
  TEMP(1,1) = 1.0
  TEMP(1,2) = X1
  TEMP(1,3) = Y1
  TEMP(1,4) = Z1
  TEMP(1,5) = X1*Y1
  TEMP(1,6) = X1*Z1
  TEMP(1,7) = X1*Y1*Z1
  TEMP(2,2) = 1.0
  TEMP(2,3) = Y1
  TEMP(2,4) = Z1
  TEMP(2,5) = Y1*Z1
  TEMP(3,3) = 1.0
  TEMP(3,4) = X1
  TEMP(3,5) = Z1
  TEMP(3,6) = X1*Z1
  TEMP(4,4) = 1.0
  TEMP(4,5) = Y1
  TEMP(4,6) = Y1
  TEMP(4,7) = X1
  TEMP(4,8) = X1*Y1

C
  DO 190 IM = 1, 8
       N(IM) = 0.0
       NSI(IM) = 0.0
       ETA(IM) = 0.0
  DO 180 JK = 1, 8
       N(IM) = N(IM) + TEMP(1,JK)*C(JK,IM)
       NSI(IM) = NSI(IM) + TEMP(2,JK)*C(JK,IM)
       ETA(IM) = ETA(IM) + TEMP(3,JK)*C(JK,IM)
       NSI(IM) = NSI(IM) + TEMP(4,JK)*C(JK,IM)
  CONTINUE
  CONTINUE
ELSE IF (ITYPE .GE. 300 .AND. IFLAG .EQ. 4) THEN

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
BHEL1 = 4

C(1,1) = 1.0
C(1,2) = XTMP(1,1)
C(1,3) = XTHP(2,1)
C(1,4) = XTHP(1,1)*XTMP(2,1)
C(2,1) = 1.0
C(2,2) = XTMP(1,2)
C(2,3) = XTHP(2,2)
C(2,4) = XTHP(1,2)*XTMP(2,2)
C(3,1) = 1.0
C(3,2) = XTMP(1,3)
C(3,3) = XTHP(2,3)
C(3,4) = XTHP(1,3)*XTMP(2,3)
C(4,1) = 1.0
C(4,2) = XTMP(1,4)
C(4,3) = XTHP(2,4)
C(4,4) = XTHP(1,4)*XTMP(2,4)

BROW = 8
BACOL = 4
EPS = 1E-20
DETER = SIMUL(BACOL, C, EPS, BROW, IOUT)

IF (DETER .EQ. 0) WRITE (*, *) 'ERROR IN SIMUL SUB-CLEB'

TEMP(1,1) = 1.0
TEMP(1,2) = X1
TEMP(1,3) = X1
TEMP(1,4) = X1

DO 200 IM = 1, 4
B(IM) = 0.0
XI(IM) = 0.0
BETA(IM) = 0.0
BSI(IM) = 0.0
B(IM) = B(IM) + TEMP(1,1)*C(1,IM)
XI(IM) = XI(IM) + TEMP(2,1)*C(1,IM)
BETA(IM) = BETA(IM) + TEMP(3,1)*C(1,IM)
BSI(IM) = BSI(IM) + TEMP(4,1)*C(1,IM)

DO 210 K = 1, BEL1
SUM = SUM + (B(IM)*AL1+BETA(IM)*AM1+BSI(IM)*AD1)*PSI(K)

210 CONTINUE

SUM = 0.0
DO 210 K = 1, BHEL1
SUM = SUM + (B(IM)*AL1+BETA(IM)*AM1+BSI(IM)*AD1)*PSI(K)

210 CONTINUE

SUM = 1.0/SUM
RETURN

END

------------------------------------------- C K I N T -------------------------------------------
SUBROUTINE CHIKIHit(L,J,PROPER,IELEM,ENEL,IDOUT,ITYPE,DTSTIF,ERROR,
$IFLAG1,IFLAG2,IFLAG3,ICTRK,ICOUP,IFLAG8)
IMPLICIT REAL*(A-H,O-Z)
C SWIT Ceggies: RE HUMB=100:10, FORMA T=900:10
C SWIT Ceggies:
C******************************************************************************
C C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C******************************************************************************
C
C******************************************************************************
C REAL*8 LTHICK
COMMON/BL0CKS/HISTO(27,70,16),CRACK(27,260,16),IHI5T,Y,IHI5T1
$ ,IHI5T2
COMMON/ABC/TESTSF(27,80,15),ITESG0,ITESG1,ITESG2
COMMON/MATER1/DEP(6,6)
COMMON/LAYHER/LTHICK(9),ZS(9),DCS(3,3),MLAYRS,HATRL,LYBUC
COMMON/ICRFIP/ICRF0,ICRF0R
DIMENSION PROPER(28),DOT(3),ROT(3,3)
REAL*8 BU3,HU3PLB,H(3,3),FT(3,3),T3XYZ(6,6),DTSTIF(6,6)
C
C******************************************************************************
C
C******************************************************************************
C C FLAG ON TO STORE CRACK ORIENT INFO FOR CRACK PLOTTING
C
C ICROF = 1
EPS1 = 0.0
EPS2 = 0.0
EPS3 = 0.0
GIF = 0.0
SIGF = 0.0
EPSHOT = 0.0
COHET = 0.0

C******************************************************************************
C SUBROUTINE TO REGISTER CRACK INITIATION PARAMETERS.
C
C INCREASE NUMBER OF CRACKS BY ONE.

C CRACK(L,1,1) = CRACK(L,1,1) + 1.

C GET NO. OF CRACKS AS AN INTEGER.

C NCRACK = INT(CRACK(L,1,1))

C RECORD THE DIRECTON COSINES OF THE CRACK NORMAL WITH RESPECT
C TO THE XYZ COORDINATE SYSTEM. (SAME AS THE MAX. PRINCIPAL
C DIRECTION).

C NCK = 9*(NCRACK-1) + 1
CRACK(L,137+NCK,J) = HISTO(L,10,J)
ROT(1,1) = HISTO(L,10,J)
CRACK(L,138+NCK,J) = HISTO(L,11,J)
ROT(1,2) = HISTO(L,11,J)
CRACK(L,139+NCK,J) = HISTO(L,12,J)
ROT(1,3) = HISTO(L,12,J)
CRACK(L,140+NCK,J) = HISTO(L,13,J)
ROT(2,1) = HISTO(L,13,J)
CRACK(L,141+NCK,J) = HISTO(L,14,J)
ROT(2,2) = HISTO(L,14,J)
CRACK(L,142+NCK,J) = HISTO(L,15,J)
ROT(2,3) = HISTO(L,15,J)
CRACK(L,143+NCK,J) = HISTO(L,16,J)
ROT(3,1) = HISTO(L,16,J)

C******************************************************************************
C
C******************************************************************************

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CRACK(L,144+BCK,J) = HISTO(L,17,J)
ROT(3,2) = HISTO(L,17,J)
CRACK(L,145+BCK,J) = HISTO(L,18,J)
ROT(3,3) = HISTO(L,18,J)

STORE THE STRESS AT WHICH THE CRACK INITIATED.

SIGMIF = DMAX1(DMAX1(HISTO(L,34,J),HISTO(L,35,J)),HISTO(L,36,J))
CRACK(L,37+BCK,J) = SIGMIF

INITIALIZE THE STATE OF CRACK TO LOADING SITUATION
I.E., INCREASE IN STRAIN.

CRACK(L,117+BCK,J) = 1.

IF THE CRACK IS RELATED TO TENSION-STIFFENING ...
IF (PROPERC16).GT.0.0) THEN
FOR THE TIME BEING SET THE TERMINATING STRAIN TO
THAT ASSIGNED EXTERNALLY.

CRACK(L,127+BCK,J) = PROPERC17)
GO TO 120
ENDIF

CALL CRACK BAND WIDTH ROUTINE TO EVALUATE THE CRACK WIDTH AT SAMPLE
POINT (BASED ON THE WORK OF OLIVER 1JNNE 1989.)

IF (DABS(ROTC1,1)).GE.0.707 .AND. DABS(ROTC1,1)).LE.0.708) THEN
ROT(1,1) = ROT(1,1) + 0.01
ROT(1,2) = ROT(1,2) - 0.01
ROT(1,3) = ROT(1,3) - 0.01
ENDIF
CALL CLEH CIELEM, L, ITYPE, HEEL, TERROR, ROT, CL)

100 CONTINUE
GF = PROPERC11)

THE FRACTURE ENERGY RELEASE AS A FUNCTION OF THE CRACK
ORIENTATION WITH RESPECT TO THE REINFORCEMENT IS ESTIMATED.

DO 110 JKL = 1, 3
IF (TENSIF(L,3+JKL,J)).GT.0.0) THEN
ISP = TENSIF(L,3+JKL,J)
ISP = 9*(ISP-1) + 1
DOTC(JKL) = TENSIF(L,30+ISP,J)*CRACK(L,137+BCK,J) + TENSIF(1
1 L,31+ISP,J)*CRACK(L,138+BCK,J) + TENSIF(L,32+ISP,J)*
2 CRACK(L,139+BCK,J))
ENDIF
110 CONTINUE
COSTH = DMAX1(DMAX1(DOTC(1),DOTC(2)),DOTC(3))
IF (TENSIF(L,4,J)).GT.0.0 .OR. TENSIF(L,6,J)).GT.0.0 .OR. TENSIF(L,6,J)
1 .GT.0) THEN
IF (DABS(COSTE) .GE. 0.866) THEN !CHECK
COFET = 1.0
ELSE
COFET = 10.0
ENDIF
ELSE
COFET = 1.0
ENDIF
GF = GF*COFET*COSTH
GF = DMX1(PROPER(11),GF)

EPSHOT = STRAIN FOR ZERO STRESS ACROSS THE CRACK.

IF (TENSTF(L,4,J).EQ.0 .AND. TENSTF(L,6,J).EQ.0 .AND. TENSTF(L,6,1).EQ.0) THEN
  EPSHOT = (18.*GF)/((SIGM1F*CL))
ELSE
  EPSHOT = GF/(SIGM1F*CL*0.9)
ENDIF

SAVE THE ULTIMATE CRACK STRAIN
CRACK(L,127+HCRACK,J) = EPSHOT

GENERATE INITIAL DIRECT STIFFNESS ACROSS THE CRACK. THIS
MUST BE EQUAL TO INFINITY USING SECANT FORMULATION. ASSU-
MING THAT THE STRESS ACROSS THE CRACK REDUCES TO 99% OF
THE INITIATION STRESS DETERMINE THE INITIAL CRACK STIFF.

120 CONTINUE
SIGCRK = .90*SIGM1F
CALL ITITES(SIGCRK, PROPER, ES, BCRACK, L, J, IDUT)
CRACK(L,67+HCRACK,J) = ES

INITIALIZE THE OLD AND NEW CRACK STRAINS.
CRACK(L,47+HCRACK,J) = .10E-10
CRACK(L,57+HCRACK,J) = .10E-10

GENERATE SHEAR STIFFNESS ALONG THE CRACK. FOR THIS
NEED THE SECANT VALUES OF 'E' & 'NU' OF CONCRETE AT
THE INITIATION OF CRACKING AND THE SHEAR RETENTION
FACTOR 'BETHA'. THE SHEAR STIFFNESS OF THE CRACK IS
ASSUMED TO REMAIN CONSTANT FOR THE ENTIRE CRACK HIST.
NOTE== THIS HAS BEEN MODIFIED TO A VARIOUS BETA MODEL BASED
ON EXPERIMENTAL DATA OF VITZIZELEOU AND TASSIOS.

ESFLR = HISTO(L,26,J)
HUSFLR = HISTO(L,27,J)
GSFLR = ESFLR/(2.*(1.+HUSFLR))
BETHA = PROPER(IO)

CRACK SHEAR STIFFNESS FOR EXPERIMENT(VITZIZELEOU)BASED SOFTENING

GCRAK1 = GSFLR*10
GCRAK2 = GSFLR*10

CRACK SHEAR STIFFNESS FOR PLAIN CONCRETE

IF (TENSTF(L,4,J).EQ.0 .AND. TENSTF(L,6,J).EQ.0 .AND. TENSTF(L,6,1).EQ.0) THEN
  BETHA = 0.10
  GCRAK1 = (BETHA*GSFLR)/(1.-BETHA)
  GCRAK2 = (BETHA*GSFLR)/(1.-BETHA)
ENDIF

CRACK(L,97+HCRACK,J) = GCRAK1
CRACK(L,107+HCRACK,J) = GCRAK2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
WRITE(*,*) 'GCRK1,GCRK1,GCRK2,GCRK2,BETHA,BETHA

FOR THE FIRST CRACK CALCULATE AND STORE THE PRINCIPAL STRAINS IN INTACT CONCRETE.

SIGH2F = HISTO(L,36,J)
SIGH3F = HISTO(L,36,J)

RETRIEVE THE CONSTITUTIVE MATRIX (IN PRINCIPAL AXES)

ES = ESFLK
HUS = HUSFLK

CONST = ES/((1.0-HUS)*(1+HUS))

DEP(1,1) = CONST*(1.0-HUS)
DEP(2,2) = CONST*(1.0-HUS)
DEP(3,3) = CONST*(1.0-HUS)
DEP(1,2) = CONST*HUS
DEP(2,1) = DEP(1,2)
DEP(3,1) = DEP(1,2)
DEP(1,3) = DEP(1,2)
DEP(2,3) = DEP(1,2)
DEP(3,2) = DEP(1,2)
DEP(4,4) = ES/(2.0*(1+HUS))
DEP(5,5) = DEP(4,4)
DEP(6,6) = DEP(4,4)

INVERT THE 'DEP' MATRIX.

H = NO. OF ROWS IN 'DEP'
HRC = MAX. NO. OF ROWS (AND COLUMNS) ALLOWED.
EPS = MIN. ALLOWABLE MAGNITUDE FOR A PIVOT ELEMENT.

ER = 6
HRC = 6
EPS = 10.0E-20

INVERSE OF 'DEP' WILL BE RETURNED IN 'DEP'.

DETER = DETERMINANT OF 'DEP' THAT IS RETURNED AS THE VALUE OF FUNCTION 'SIMUL'.

DETER = SIMUL(NR,DEP,EPS,HRC,IOUT)
IF (DETER .LT. 0.0) THEN
  WRITE (IOUT, 900) L, J, IELEM
STOP
ENDIF

EPS1 = DEP(1,1)*SIGH1F + DEP(1,2)*SIGH2F + DEP(1,3)*SIGH3F
EPS2 = DEP(2,1)*SIGH1F + DEP(2,2)*SIGH2F + DEP(2,3)*SIGH3F
EPS3 = DEP(3,1)*SIGH1F + DEP(3,2)*SIGH2F + DEP(3,3)*SIGH3F
AEMAX = DMAX1(DMAX1(DABS(EPS1),DABS(EPS2)),DABS(EPS3))
IF (DABS(EPS1) .LE. 1E-3*AEMAX) EPS1 = 0.0
IF (DABS(EPS2) .LE. 1E-3*AEMAX) EPS2 = 0.0
IF (DABS(EPS3) .LE. 1E-3*AEMAX) EPS3 = 0.0
IF (HCRACK .LT. 1) HISTO(L,37,J) = DMAX1(DMAX1(EPS1,EPS2),EPS3)

TENSION STIFFENING IN STEEL DIRECTION

IFLAG1 = 0
IFLAG2 = 0
IFLAG3 = 0
IFLAG4 = 0
IFLAG5 = 0
IFLAG6 = 0
IMARK1 = 0
IMARK2 = 0
IMARK3 = 0
MCHECK = 0
DO 180 ISP = 1, 3
C CHECK THE DIRECTION OF THE CRACK WITH AN EXISTING TENSION-STIFFENING
C DIRECTION FOR(SECONDARY CRACKS). IF DIRECTION OF THIS CRACK NORMAL IS
C CLOSE TO THAT OF THE STEEL THEN NO STRAIN SOFTENING CONSIDERED. ELSE
C THIS IS A STRAIN SOFTENING CRACK AND MUST BE TREATED AS SUCH.
C
C IF (TENSTF(L,ISP,J) .EQ. 1.0) THEN
IF (ISP .EQ. 1) IFLAG1 = 1
IF (ISP .EQ. 2) IFLAG2 = 1
IF (ISP .EQ. 3) IFLAG3 = 1
ISPG = INT(TENSTF(L,2+ISP,J))
ISHG = (ISPG-1)*9 + 1
C
C ... TENSION STIFFENING (STEEL) DIRECTION W.R.T TO 'GLOBAL'
C
N(1,1) = TENSTF(L,15+30,J)
N(1,2) = TENSTF(L,15+31,J)
N(1,3) = TENSTF(L,15+32,J)
N(2,1) = TENSTF(L,15+33,J)
N(2,2) = TENSTF(L,15+34,J)
N(2,3) = TENSTF(L,15+35,J)
N(3,1) = TENSTF(L,15+36,J)
N(3,2) = TENSTF(L,15+37,J)
N(3,3) = TENSTF(L,15+38,J)
C
C ... CRACK NORMAL W.R.T TO 'GLOBAL'
C
NT(1,1) = HIST0(L,10,J)
NT(1,2) = HIST0(L,11,J)
NT(1,3) = HIST0(L,12,J)
NT(2,1) = HIST0(L,13,J)
NT(2,2) = HIST0(L,14,J)
NT(2,3) = HIST0(L,15,J)
NT(3,1) = HIST0(L,16,J)
NT(3,2) = HIST0(L,17,J)
NT(3,3) = HIST0(L,18,J)
C
RDOT = N(1,1)*ET(1,1) + N(1,2)*ET(2,1) + N(1,3)*ET(3,1)
IF (DABS(RDOT) .LT. 0.906) MCHECK = 1
GO TO 170
ENDIF
C
C SEE IF TENSION STIFFENING IS ACTIVATED FOR THE CRACK.
C
C IF (TENSTF(L,ISP,J) .EQ. 0.0) THEN
ISPG = INT(TENSTF(L,5+ISP,J))
IF (ISP .EQ. 0) THEN
IF (ISP .EQ. 1) IFLAG1 = 1
IF (ISP .EQ. 2) IFLAG2 = 1
IF (ISP .EQ. 3) IFLAG3 = 1
GO TO 170
ENDIF
C
C ... TENSION STIFFENING (STEEL) DIRECTION W.R.T TO 'GLOBAL'
c
n(1,1) = testst(l,ismg+30,j)
n(1,2) = testst(l,ismg+31,j)
n(1,3) = testst(l,ismg+32,j)
n(2,1) = testst(l,ismg+33,j)
n(2,2) = testst(l,ismg+34,j)
n(2,3) = testst(l,ismg+35,j)
n(3,1) = testst(l,ismg+36,j)
n(3,2) = testst(l,ismg+37,j)
n(3,3) = testst(l,ismg+38,j)

C ... CRACK NORMAL W.R.T TO 'GLOBAL'

nt(1,1) = hist0(l,10,j)
n(2,1) = hist0(l,11,j)
n(3,1) = hist0(l,12,j)
n(1,2) = hist0(l,13,j)
n(2,2) = hist0(l,14,j)
n(3,2) = hist0(l,15,j)
n(1,3) = hist0(l,16,j)
n(2,3) = hist0(l,17,j)
n(3,3) = hist0(l,18,j)

C
root = n(1,1)*nt(1,1)+n(1,2)*nt(2,1)+n(1,3)*nt(3,1)
C
C ... FIND THE ORIENTATION OF THE CRACK WITH RESPECT TO STEEL AND
C INITIATE A TENS. STIFF SPRING IF NECESSARY.

C
do 140 imm = 1, 3
   do 130 imp = 1, 3
      rot(imm,imp) = 0.0
      rot(imm,imp) = rot(imm,imp) + n(imm,1)*nt(1,imm,imp)
      1
      rot(imm,imp) = rot(imm,imp) + n(imm,2)*nt(2,imm,imp)
      1
      rot(imm,imp) = rot(imm,imp) + n(imm,3)*nt(3,imm,imp)
   130 continue
140 continue

C
do 88 km = 1, 3
   do 87 km = 1, 3
      if (dabs(rot(kk,km)) .le. 0.01) rot(kk,km) = 0.0
   end do
   end do
C
C ... STRAIN IN TENSIONSTIFFENING (STEEL) DIRECTION IS...

C
epsstl = eps1*rot(1,1)*rot(1,1) + eps2*rot(1,2)*rot(1,2)
  + eps3*rot(1,3)*rot(1,3)

C
if (dabs(epsstl) .lt. 1.0e-9) epsstl = 0.0
if (isp .eq. 1) imark1 = 1
if (isp .eq. 2) imark2 = 1
if (isp .eq. 3) imark3 = 1
if (dabs(rot).le.0.025) and. epsstl.lt.0.0) then
   if (isp .eq. 1) imark1 = 0
   if (isp .eq. 2) imark2 = 0
   if (isp .eq. 3) imark3 = 0
   if (isp .eq. 1) iflag4 = 2
   if (isp .eq. 2) iflags = 2

304
IF (ISP .EQ. 3) IFLAG6 = 2
ENDIF

IF (DABS(RDOT) .GE. 0.25) THEN
THEB
C
ACTIVATE TENSIONSTIFFENING SPRINGS WHEN STRAINS IN THE SPRING DIR.
C
AT CRACKING EXCEED U N I A X I A L CRACKING STRAINS.
CKSTIB = PROPER(6)/PROPER(1)
IF (EPSSTL .GE. CKSTIB) THEN
HCHECK = 0
IF (ISP .EQ. 1) IFLAG1 = 1
IF (ISP .EQ. 2) IFLAG2 = 1
IF (ISP .EQ. 3) IFLAG3 = 1
TENSTF(L,ISP,J) = 1.0
ELSE IF (EPSSTL .GE. 0.0 AND EPSSTL .LT. CKSTIB)
THEN
IF (ISP .EQ. 1) IFLAG4 = 2
IF (ISP .EQ. 2) IFLAG5 = 2
IF (ISP .EQ. 3) IFLAG6 = 2
ENDIF
C
STORE THE CRACKING STRAIN AT U N I A X I A L CRACKING STRAINS USED AS A
C PARAMETER.
C
IF (CKSTIB .LE. 0.0) THEN
TENSTF(L,9+ISPG,J) = 0.0
ELSE
TENSTF(L,9+ISPG,J) = CKSTIB
ENDIF
C
AL1 = ROT(1,1)
AM1 = ROT(2,1)
AH1 = ROT(3,1)
AL2 = ROT(1,2)
AM2 = ROT(2,2)
AH2 = ROT(3,2)
AL3 = ROT(1,3)
AM3 = ROT(2,3)
AH3 = ROT(3,3)
C
CHECK FOR SPECIAL CASES.....
C
IF (TENSTF(L,4,J).EQ.0 OR TENSTF(L,5,J).EQ.0 OR TEHSTF(L,6,J).EQ.0)
THEN
IF (DABS(AB3) .LE. 0.01) THEN
AL1 = ROT(1,1)
AL2 = ROT(1,3)
AL3 = ROT(1,2)
AM1 = ROT(2,1)
AM2 = ROT(2,3)
AM3 = ROT(2,2)
AH1 = ROT(3,1)
AH2 = ROT(3,2)
AH3 = ROT(3,3)
ENDIF
C
C CHECK FOR SPECIAL CASES.....
C
IF (DABS(AB3) .LE. 0.01) WRITE (* , *) 'AB3', AB3
IF (DABS(AL1) .LE. 0.01) WRITE (*, *) 'AL1', AL1
IF (DABS(AM1) .LE. 0.01) WRITE (* , *) 'AM1', AM1
C
ENDIF
IF (TENSFL(L,4,J).EQ.0 .AND. TENSFL(L,5,J).EQ.0 .AND. TENSFL(L,6,J).EQ.0 .AND. TENSFL(L,4,J).EQ.0 .AND. TENSFL(L,5,J).EQ.0 .AND. TENSFL(L,6,J).EQ.0) THEN
IF (DABS(AM2) .LE. 0.01) THEN
   AL1 = ROT(1,1)
   AL2 = ROT(1,3)
   AL3 = ROT(1,2)
   AM1 = ROT(2,1)
   AM2 = ROT(2,3)
   AM3 = ROT(2,2)
   AH1 = ROT(3,1)
   AH2 = ROT(3,3)
   AH3 = ROT(3,2)
ENDIF
IF (DABS(AM2) .LE. 0.01) WRITE (* , *) 'AM2', 'AH1', 'AH2', 'AH3', 'AL1', 'AL2', 'AL3'
ENDIF
C EVALUATE THE SPRING INITIATING STRESSES AND STORE THEM HERE IN CASE C IT IS DETERMINED TO BE ACTIVATED ALONG WITH ANOTHER SPRING DIRECTION C IN THIS STEP.
C IF (TENSFL(L,4,J).GT.0 .0 .AND. TENSFL(L,5,J).GT.0 .0 .AND. TENSFL(L,6,J).GT.0 .0) THEN
   C ... ALL THREE STEEL DIRECTIONS PRESENT (NEED NOT BE ACTIVE).
   TENSFL(L,6+ISPG,J) = SIGMIF
   WRITE(* , *) 'THREE SIGMIF', SIGMIF, 'SIGMIF', SIGMIF
   ELSE IF (TENSFL(L,4,J).GT.0 .0 .AND. TENSFL(L,5,J).EQ.0 .0 .AND. TENSFL(L,6,J).GT.0 .0 .AND. TENSFL(L,4,J).EQ.0 .0 .OR. TENSFL(L,5,J).EQ.0 .0 .AND. TENSFL(L,6,J).EQ.0 .0 .AND. TENSFL(L,4,J).GT.0 .0 .OR. TENSFL(L,5,J).EQ.0 .0 .AND. TENSFL(L,6,J).EQ.0 .0 .AND. TENSFL(L,4,J).GT.0 .0) THEN
   C ... ANY TWO DIRECTIONS PRESENT (NEED NOT BE ACTIVE).
   SIGF = SIGMIF*((AL3*AH1-AL1*AH3)*(AM2*AH3-AM3*AH2)-(AM1*AH2-AM2*AH1)*(AL1*AH2-AL3*AH1-(AM2*AH3-AM3*AH2)))/((AL3*AH3-AL1*AH1-AL2)*(AM2*AH3-AM3*AH2-AL1)-(AM2*AH3-AM3*AH1))
   WRITE(* , *) 'ANY TWO SIGF', SIGF, 'SIGMIF', SIGMIF
   TENSFL(L,6+ISPG,J) = SIGF
   ELSE IF (TENSFL(L,4,J).GT.0 .0 .AND. TENSFL(L,5,J).EQ.0 .0 .AND. TENSFL(L,6,J).EQ.0 .0 .AND. TENSFL(L,4,J).EQ.0 .0 .OR. TENSFL(L,5,J).EQ.0 .0 .AND. TENSFL(L,6,J).EQ.0 .0 .AND. TENSFL(L,4,J).EQ.0 .0 .OR. TENSFL(L,5,J).EQ.0 .0 .AND. TENSFL(L,6,J).EQ.0 .0) THEN
   C ... ONLY ONE DIRECTION PRESENT (NEED NOT BE ACTIVE).
   SIGF = SIGMIF*(((AM1+AM2-AM1+AM2)*(AL3+AM3-AM3+AM2)*(AL2+AM2-AM2+AL3)-(AM2+AM3-AM3*AM2)))/((AL1+AM2+AM3+AM2-AM3+AM3-AM3*AM2))
   WRITE(* , *) 'ONE ONLY SIGF', SIGF, 'SIGMIF', SIGMIF
   TENSFL(L,6+ISPG,J) = SIGF
C SET THE SPRING INITIATING STRAINS HERE FOR CONTINUOUS MONITORING DURING THE LOADING PROCEDURE.
C

IF (EPSSTL .LE. 0.0) THEN
  TEHSTF(L,18+ISPG,J) = 0.0
  TEHSTF(L,18+ISPG,J) = 0.0
ELSE IF (EPSSTL .GE. CKSTIN) THEN
  TEHSTF(L,18+ISPG,J) = CKSTIN
  TEHSTF(L,18+ISPG,J) = CKSTIN
ELSE
  TEHSTF(L,18+ISPG,J) = EPSSTL
  TEHSTF(L,18+ISPG,J) = EPSSTL
ENDIF

C SET THE SPRING INITIATING STIFFNESS HERE FOR POSSIBLE ACTIVATION DURING THE LOADING PROCEDURE.
C

IF (EPSSTL .GE. CKSTIN) THEN
  TEHSTF(L,ISPG+24,J) = SIGF/CKSTIN
ELSE IF (EPSSTL .LT. CKSTIN .AND. EPSSTL .GT. 0.0) THEN
  TEHSTF(L,ISPG+24,J) = SIGF/EPSSTL
ENDIF

C SET THE SPRING INITIATING STATE TO LOADING
C
  TEHSTF(L,21+ISPG,J) = 1.0
ENDIF

C ... IF THE CURRENT CRACK IS NOT STRAIN SOFTENING THEN CHECK IF A TENSION STIFFENING DIRECTION IS ACTIVATED AT THIS STAGE ELSE CONTINUE STRAIN SOFTENING UNTIL THE STRAINS PICK UP FOR THE ACTIVATION OF TENSION STIFFENING
C
WRITE(*, 'CKINIT FLAG CKR,CRA(C(L,67+HCRACK,J))')
WRITE(*, 'TFG1,IFLAG1,TFG2,IFLAG2,TFG3,IFLAG3')
WRITE(*, 'TFG4,IFLAG4,TFG5,IFLAG5,TFG6,IFLAG6')
IF (MCHECK .EQ. 0) THEN
  IF (IFLAG1.EQ.0 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.1) THEN
    ITCRK = HCRACK
  ELSE IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.0 .OR.
     1 IFLAG1.EQ.1 .AND. IFLAG3.EQ.1 .AND. IFLAG2.EQ.0 .OR.
     2 IFLAG2.EQ.1 .AND. IFLAG3.EQ.1 .AND. IFLAG1.EQ.0) THEN
    IF (IFLAG4.EQ.1 .AND. IFLAG6.EQ.1) THEN
      ITCRK = HCRACK
    ELSE IF (IFLAG4.EQ.2 .AND. IFLAG5.EQ.1) THEN
      TEHSTF(L,1,J) = 1.0
    ELSE IF (IFLAG5.EQ.2 .AND. IMARK2.EQ.1) THEN
      TEHSTF(L,2,J) = 1.0
    ELSE IF (IFLAG6.EQ.2 .AND. IMARK3.EQ.1) THEN
      TEHSTF(L,3,J) = 1.0
    ENDIF
    IFLAG1 = 1
    IFLAG2 = 1
ENDIF

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IFLAG3 = 1
ENDIF
ELSE IF (TESTF(L,4,J)).EQ.0.0 .OR. TESTF(L,5,J)).EQ.0.0
1 .OR. TESTF(L,6,J)).EQ.0.0) THEN
ITCRK = CRACK
IFLAG1 = 1
IFLAG2 = 1
IFLAG3 = 1
ENDIF
ELSE IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.0 .AND. IFLAG3.EQ.0 .OR.
1 IFLAG2.EQ.1 .AND. IFLAG3.EQ.0 .AND. IFLAG1.EQ.0 .OR.
2 IFLAG3.EQ.1 .AND. IFLAG1.EQ.0 .AND. IFLAG2.EQ.0) THEN
1 IF (IFLAG4.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.1) THEN
2 .AND. TESTF(L,4,J).GT.0.0) THEN
3 IF (IFLAG4.EQ.2) TESTF(L,1,J) = 1.0
4 IF (IFLAG5.EQ.2) TESTF(L,2,J) = 1.0
5 IF (IFLAG6.EQ.2) TESTF(L,3,J) = 1.0
6 ITCRK = CRACK
7 IFLAG1 = 1
8 IFLAG2 = 1
9 IFLAG3 = 1
ENDIF
ELSE IF (TESTF(L,4,J)).EQ.0.0 .AND. TESTF(L,5,J)).EQ.0.0
1 .AND. TESTF(L,6,J)).EQ.0.0 .OR. TESTF(L,5,J)).EQ.0.0 .OR. TESTF(L,6
2 ).EQ.0.0) THEN
3 ITCRK = CRACK
4 IFLAG1 = 1
5 IFLAG2 = 1
6 IFLAG3 = 1
ENDIF
ENDIF
C IF(ELSE.EQ.1) THEN
C WRITE(*,*) 'CRICT EXIT STIF',CRACK(CRACK,L=6+CRACK,J)
C END IF
C END IF
1 DO 200 IH = 1, 6
2 DTSTIF(CRACK,ISPC) = 0.0
3 CONTINUE
4 DO 250 ISP = 1, 3
5 ISP = TESTF(L,3+ISP,J)
6 IF (ISP.GT.0 .AND. TESTF(L,ISP,J)).EQ.1.0) THEN
7 ISMS = (ISP-1)*9 + 1
8 C ... TENSION STIFFNESS (STEEL) DIRECTION W.R.T TO 'GLOBAL'
C
9 N(1,1) = TESTF(B,ISMS+30,J)
10 N(1,2) = TESTF(B,ISMS+31,J)
11 N(1,3) = TESTF(B,ISMS+32,J)
12 N(2,1) = TESTF(B,ISMS+33,J)
13 N(2,2) = TESTF(B,ISMS+34,J)
14 N(2,3) = TESTF(B,ISMS+35,J)
C

C******************************************************************************
C
C INCLUDE(PROCESS)
C
C SUBROUTINE CRKPCT(EPSX,EPSY,EPSZ,GAYAX,GAYAY,GAYAZ,PROPER,
$ TOLERE,CHKGUS,RSTIFF,CRKCOH,OVERFT,SIGMA,SIGMA2,SIGMA3,LJ,PDIR,LAYERS,\n$ MAXCK2,SIGHA(6),DCHV(30,30))
C
C IMPLICIT REAL*8 (A-H,O-Z)
C
C******************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DMATS PSTSST PSTST VITAS ARANGE CRACKE CRACOS
C OVERFT SIMUL
C
C******************************************************************************
C
INTEGER ELHUM
REAL*8 LTHICK

COMMON/BLOCS/HISTO(27,70,15),CRACK(27,250,15),HISTY,HIST1
$,HIST2
COMMON/ABC/TEKSTF(27,80,16),IT6SPG,IT6SG1,IT6SG2
COMMON/MATER/DEP(6,6)
COMMON/LAYERS/LTHICK(9),ZS(9),DCS(3,3),MLAYRS,MATRL,LYHUM
DIMENSION PROPER(28),SIGMA(6),DC6(30,30)
REAL*8 BUS,HT(3,6),FVAL(3),PDIR(3,3),DS(3,3),DST(3,3)
LOGICAL CHKGUS

C******************************************************************************
C
C SUBROUTINE TO CHECK STATUS AT A CRACKED POINT AND
C FINALLY RETURN THE PRINCIPAL STRESSES OF INTACT CONCRETE AND TOTAL
C STRESSES FOR FURTHER INVESTIGATION.
C
C******************************************************************************
ASSUME A SINGLE CRACK AND INTACT CONCRETE CONNECTED IN SERIES AND UNDER UNIAXIAL TENSION. THE EQUIVALENT STIFFNESS OF THIS SYSTEM IS VERY CLOSE TO THE STIFFNESS OF THE INTACT CONCRETE FOR CRACK STIFFNESSES MANY TIMES LARGER THAN THE INTACT CONCRETE MODULUS OF ELASTICITY. THEREFORE, THE SELECTED TOLERANCE WILL PRACTICALLY BE EFFECTIVE FOR CRACK STIFFNESSES SMALLER THAN THE INTACT CONCRETE STIFFNESS.

... TOLERE = TOLERANCE USED FOR INTACT CONCRETE

TOLER = TOLERE
ISOFT = 0
ITCRK = 0

.... RETRIEVE THE GLOBAL COMPOSITE CONSTITUTIVE MATRIX.

ICODE = 1
IPG = 0
CALL DRATS2 (ELNUM, ITYPE, L, IFLAGS, IDOUT, ICODE, IPG)
D11 = DEP(1,1)
D12 = DEP(1,2)
D13 = DEP(1,3)
D14 = DEP(1,4)
D16 = DEP(1,6)
D21 = DEP(2,1)
D22 = DEP(2,2)
D23 = DEP(2,3)
D24 = DEP(2,4)
D26 = DEP(2,6)
D28 = DEP(2,8)
D29 = DEP(2,9)
D31 = DEP(3,1)
D32 = DEP(3,2)
D33 = DEP(3,3)
D34 = DEP(3,4)
D35 = DEP(3,5)
D36 = DEP(3,6)
D41 = DEP(4,1)
D42 = DEP(4,2)
D43 = DEP(4,3)
D44 = DEP(4,4)
D46 = DEP(4,6)
D49 = DEP(4,9)
D51 = DEP(5,1)
D52 = DEP(5,2)
D53 = DEP(5,3)
D54 = DEP(5,4)
D56 = DEP(5,6)
D61 = DEP(6,1)
D62 = DEP(6,2)
D63 = DEP(6,3)
D64 = DEP(6,4)
D65 = DEP(6,5)
D66 = DEP(6,6)

HAVING THE GLOBAL STRAINS DETERMINE GLOBAL STRESSES.

.... REDUCE TRUNCATION ERRORS DURING ADDITION/SUBTRACTION.
XYZ = DMAX1(DMAX1(DMAX1(DMAX1(DHAX1(DHAX1(DHAX1(DHAX1(DABS(EPSX),DABS(EPSY),DABS(GAMAYX),DABS(GAMAYZ),DABS(GAMAXY),DABS(GAMAXZ),DABS(EPSZ)))))))

IF (DABS(EPSX) .LT. XYZ*10.0E-7) EPSX = 0.0
IF (DABS(EPSY) .LT. XYZ*10.0E-7) EPSY = 0.0
IF (DABS(GAMAYX) .LT. XYZ*10.0E-7) GAMAYX = 0.0
IF (DABS(GAMAYZ) .LT. XYZ*10.0E-7) GAMAYZ = 0.0
IF (DABS(GAMAXY) .LT. XYZ*10.0E-7) GAMAXY = 0.0
IF (DABS(GAMAXZ) .LT. XYZ*10.0E-7) GAMAXZ = 0.0

SIGMA(1) = D11*EPSX + D12*EPSY + D14*GAMAYX + D13*EPSZ
SIGMA(2) = D21*EPSX + D22*EPSY + D24*GAMAXY + D23*EPSZ
SIGMA(3) = D31*EPSX + D32*EPSY + D34*GAMAXZ + D33*EPSZ
SIGMA(4) = D41*EPSX + D42*EPSY + D44*GAMAYZ + D43*EPSZ
SIGMA(5) = D51*EPSX + D52*EPSY + D54*GAMAXZ + D53*EPSZ
SIGMA(6) = D61*EPSX + D62*EPSY + D64*GAMAYZ + D63*EPSZ

XYZ = DMAX1(DMAX1(DMAX1(DMAX1(DHAX1(DHAX1(DHAX1(DABS(SIGMA(1)) .DABS(SIGMA(2) ) ) . DABS (SIGMA (3 ) ) ) . DABS (SIGMA (4) ) ) . DABS (SIGMA (B) ) ) . DABS (SIGMA (6) ) ) . LT. XYZ*10.0E-4) SIGMA

IF (DABS(SIGMA(1)) .LT. XYZ*10.0E-4) SIGMA(1) = 0.0
IF (DABS(SIGMA(2)) .LT. XYZ*10.0E-4) SIGMA(2) = 0.0
IF (DABS(SIGMA(3)) .LT. XYZ*10.0E-4) SIGMA(3) = 0.0
IF (DABS(SIGMA(4)) .LT. XYZ*10.0E-4) SIGMA(4) = 0.0
IF (DABS(SIGMA(5)) .LT. XYZ*10.0E-4) SIGMA(5) = 0.0
IF (DABS(SIGMA(6)) .LT. XYZ*10.0E-4) SIGMA(6) = 0.0

IFG = 2

DETERMINE NUMBER OF CRACKS AT THIS POINT.
BCRACK = INT(CRACK(L,1,J))
DO 160 ICRACK = 1, BCRACK

DETERMINE TRANSPOSE OF THE TRANSFORMATION MATRIX 'H'.
ICK = 9*ICRACK-1 + 1
AL1 = CRACK(L,137+ICK,J)
AM1 = CRACK(L,138+ICK,J)
AB1 = CRACK(L,139+ICK,J)
AL2 = CRACK(L,140+ICK,J)
AM2 = CRACK(L,141+ICK,J)
AL3 = CRACK(L,142+ICK,J)
AM3 = CRACK(L,143+ICK,J)
AN3 = CRACK(L,146+ICK,J)

BT(1,1) = AL1*AL1
BT(1,2) = AM1*AM1
BT(1,3) = AB1*AB1
BT(1,4) = 2.0*AL1*AM1
BT(1,5) = 2.0*AM1*AB1
BT(1,6) = 2.0*AB1*AL1
BT(2,1) = AL1*AL2
BT(2,2) = AM1*AM2
BT(2,3) = AB1*AB2
BT(2,4) = AL1*AM2 + AM1*AL2
BT(2,5) = AM1*AM2 + AB1*AB2
BT(2,6) = AM1*AL2 + AL1*AM2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CALCULATE CRACK INTERFACE STRESSES.

SIGCRK = DIRECT STRESS ACROSS THE CRACK.
TAUCRK = SHEAR ALONG .

SIGCRK = 0.0
TAUCRK1 = 0.0
TAUCRK2 = 0.0

DO 110 JJ = 1, 3
  DO 100 II = 1, 6
    IF (II .EQ. 1) THEN
      SIGCRK = SIGCRK + HT(II, JJ) * SIGMA(JJ)
    ELSE IF (II .EQ. 2) THEN
      TAUCRK1 = TAUCRK1 + HT(II, JJ) * SIGMA(JJ)
    ELSE
      TAUCRK2 = TAUCRK2 + HT(II, JJ) * SIGMA(JJ)
    ENDIF
  CONTINUE
110 CONTINUE

IF (DABS(TAUCRK1) .LT. DABS(0.01 * PROPER(6))) TAUCRK1 = 0.0
IF (DABS(TAUCRK2) .LT. DABS(0.01 * PROPER(6))) TAUCRK2 = 0.0

D CRACK = CRACK(L, 87 + ICRACK, J)
GCRCK1 = CRACK(L, 97 + ICRACK, J)
GCRCK2 = CRACK(L, 107 + ICRACK, J)

DETERMINE CRACK INTERFACE STRAINS.

NO COUPLING BETWEEN CRACK'S DIRECT AND SHEAR STIFFNESSES; THEREFORE,

EPSCKR = SIGCRK / DCRACK
GMCKR1 = TAUCRK1 / GCRCK1
GMCKR2 = TAUCRK2 / GCRCK2

C -PRIH. STRESSES NEEDED FOR SHEAR STIF. CALCULATIONS.
CALL PSTSH(SIGMA, PVAL, PDIR, IFG, IOUT)
SIGMA1 = PVAL(1)
SIGMA2 = DMAX1(PROPER(6), SIGMA1)
SIGMA3 = PVAL(2)
SMAX = DMAX1(DMAX1(DABS(SIGMA1), DABS(SIGMA2)), DABS(SIGMA3))
IF (DABS(SIGMA1) .LE. SMAX * 1.E-4) SIGMA1 = 0.0
IF (DABS(SIGMA2) .LE. SMAX * 1.E-4) SIGMA2 = 0.0
IF (DABS(SIGMA3) .LE. SMAX * 1.E-4) SIGMA3 = 0.0
CALL VISTAS(GMCKR1, GMCKR2, EPSCKR, SIGMA1, SIGMA2, SIGMA3, PDIR, ICRACK, L, IOUT, ELNUM, HSEL, LAYERS)

RETRIEVE THE CRACK'S PREVIOUS STRAIN FOR COMPARISON.
EPSOLD = CRACK(L, 47 + ICRACK, J)

DETERMINE THE STATE OF THE CRACK AT THE END OF
THE PREVIOUS ITERATION.

STATE = CRACK(L,117+ICRACK,J)

... DUGDALE MODEL FOR REINFORCED CRACKED POINTS PRIOR TO THE INITIATION OF TENSION STIFFENING.

EPSSLT = CRACK(L,127+ICRACK,J)
AREA = CRACK(L,37+ICRACK,J)+EPSSLT+0.9
SUBAREA = 0.5*CRACK(L,37+ICRACK,J)*(EPSCRK-EPSSLT)

CHECK FOR CONVERGENCE ON THE DUGDALE ENVELOPE.

IF (TENSTF(L,4,J).GT.0.0 OR. TENSTF(L,6,J).GT.0.0 OR. TENSTF(L,6,J).GT.0.0) THEN
  IFLAG1 = 0
  IFLAG2 = 0
  IFLAGB = 0
  DO 120 ISP = 1,3

  CHECK THE DIRECTION OF THE CRACK WITH AN EXISTING STEEL DIRECTION FOR(SECONDARY CRACKS). IF DIRECTION OF THIS CRACK NORMAL IS CLOSE TO THAT OF THE STEEL THEN NO STRAIN SOFTENING CONSIDERED. ELSE THIS IS A STRAIN SOFTENING CRACK AND MUST BE TREATED AS SUCH.

  IF (TENSTF(L,ISP+3,J).NE.0.0) THEN
    ISP0 = ISP(TENSTF(L,ISP+3,J))
    ISMG = (ISPG-1)+1

  TENSION STIFFENING (STEEL) DIRECTION W.R.T TO 'GLOBAL'

  DB(1,1) = TENSTF(L,ISMG+30,J)
  DB(1,2) = TENSTF(L,ISMG+31,J)
  DB(1,3) = TENSTF(L,ISMG+32,J)
  DB(2,1) = TENSTF(L,ISMG+33,J)
  DB(2,2) = TENSTF(L,ISMG+34,J)
  DB(2,3) = TENSTF(L,ISMG+35,J)
  DB(3,1) = TENSTF(L,ISMG+36,J)
  DB(3,2) = TENSTF(L,ISMG+37,J)
  DB(3,3) = TENSTF(L,ISMG+38,J)

  CRACK NORMAL W.R.T TO 'GLOBAL'

  ICK = 9*(ICRACK-1) + 1
  DNT(1,1) = CRACK(L,137+ICK,J)
  DNT(2,1) = CRACK(L,138+ICK,J)
  DNT(3,1) = CRACK(L,139+ICK,J)
  DNT(1,2) = CRACK(L,140+ICK,J)
  DNT(2,2) = CRACK(L,141+ICK,J)
  DNT(3,2) = CRACK(L,142+ICK,J)
  DNT(1,3) = CRACK(L,143+ICK,J)
  DNT(2,3) = CRACK(L,144+ICK,J)
  DNT(3,3) = CRACK(L,145+ICK,J)

  RDOT = DB(1,1)*DNT(1,1) + DB(1,2)*DNT(2,1) + DB(1,3)*DNT(3,1)
  PI = 4.0*DAN(1.0DO)
  IF (DABS(RDOT).LT. DCOS((46.0*PI)/180.0)) THEN
    IF (ISP .EQ. 1) IFLAG1 = 1
    IF (ISP .EQ. 2) IFLAG2 = 1
    IF (ISP .EQ. 3) IFLAG3 = 1
  ENDIF

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ELSE IF (TENSTF(L,ISP+3,J),EQ,0) THEN
  IF (ISP,EQ,1) IFLAG1 = 1
  IF (ISP,EQ,2) IFLAG2 = 1
  IF (ISP,EQ,3) IFLAG3 = 1
ENDIF
120 CONTINUE
1 IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.1)
  GO TO 130
C
IF (EPSCRK.GE.EPSULT) THEN
  IF (TENSTF(L,1,J),EQ,0 .0 .AND. TENSTF(L,2,J),EQ,0 .0
  .AND. TENSTF(L,3,J),EQ,0 .0 ) THEN
    WRITE (* , *)
    'WARNING STRAIN SOFTENING LIMIT FOR CRACK',
    'FRACTURE ENERGY FULLY DESSIPATED', ICRACK,
    'GAUSS PT', L, 'LAYER', J, 'ELEM', ELNUM
    ITCRK = ICRACK
    ISOFT = 1
  ELSE.
    CRACK(L,87+ICRACK,J) = PROPER(1)/1000.0
  ENDIF
ELSE IF (EPSCRK.LT.0 .0 .AND. HCRACK.GT.1 .0 ) THEN
  CALL ARAUGE (ICRACK, HCRACK, L, J)
ENDIF
CRACK(L,87+ICRACK,J) = MAX1 (PROPER(1)/1000.0 , CRACK(L,87+ICRACK,J))
1 IF (CRACK(L,87+ICRACK,J),LE. PROPER(1)/1000.0) THEN
  CRACK(L,77+ICRACK,J) = GMCRK1
  CRACK(L,47+ICRACK,J) = GMCRK2
  CRACK(L,57+ICRACK,J) = EPSCRK
  GO TO 140
ENDIF
CI = (SUBARA/AREA)+100.0
1 IF (CI .GE. TOLER) THEN
  CSTIFF = (CRACK(L,37+ICRACK,J)*(1.0-0.2*EPSCRK/EPSULT))/EPSCRK
  IF (CSTIFF.LT.0 .0 .AND. CRACK(L,37+ICRACK,J).LT.0 .0 ) THEN
    CRACK(L,37+ICRACK,J) = CSTIFF
    CSTIFF = 0.0
    CHGUS = .TRUE.
    HSTIFF = 1
    CONVER = 999.0
  ENDIF
ELSE
  C
  CHECK FOR OVERSHOOTING OF THE STRESS POINT.
  C
  IF (SIGCRK .GT. CRACK(L,37+ICRACK,J)) THEN
    EPSCRK = EPSOLD + TOLER*AREA/(0.5+CRACK(L,37+
    1 ICRACK,J)*100.)
    CSTIFF = (CRACK(L,37+ICRACK,J)*(1.0-0.2*EPSCRK/
    1 EPSULT))/EPSCRK
    IF (CSTIFF.LT.0 .0 .AND. CRACK(L,37+ICRACK,J).LT.0 .0 ) THEN
      CRACK(L,37+ICRACK,J) = CSTIFF
      CSTIFF = 0.0
      CONVER = 999.0
      HSTIFF = 1
      CHGUS = .TRUE.
    ENDIF
  ENDIF
ENDIF
CRACK(L,87+ICRACK,J) = MAX1 (PROPER(1)/1000.0 , CRACK(L,87+ICRACK,J))
I CRACK, J)
CRACK(L, 67 + I CRACK, J) = G M CR K1
CRACK(L, 77 + I CRACK, J) = G M CR K2
CRACK(L, 47 + I CRACK, J) = E P S CRK
CRACK(L, 57 + I CRACK, J) = E P S CRK
GO TO 140
ENDIF

THE FOLLOWING IS FOR PURE STRAIN SOFTENING CRACKS ONLY.

--------------------
|                    |
|                       |
|                      |
--------------------

ENDF

THE FOLLOWING IS FOR PURE STRAIN SOFTENING CRACKS ONLY.

----------------------
|                     |
| OPENING CRACK       |
|                     |
----------------------

130 CONTINUE

IF (EPS CRK.GT.0.0 .AND. STATE.EQ.4.) THEN

A CLOSED CRACK STARTS TO OPEN. USE 'EPSOLD' TO
DETERMINE A NEW CROSS-CRACK STIFFNESS.

CALL CRACK(EPSOLD, PROPER, ES, I CRACK, L, J)
EINITL = PROPER(1)
ES = DMAX1(EINITL/1000., 0, ES)
CRACK(L, 67 + I CRACK, J) = ES

FOR THE CURRENT ITERATION ASSUME THAT THE CRACK OPENS
UP TO ZERO STRAIN.

CRACK(L, 57 + I CRACK, J) = 0.0

SET THE STATE OF THE CRACK TO RELOADING.

CRACK(L, 117 + I CRACK, J) = 3.

UPDATE THE CONSTITUTIVE LAW, UPDATE THE CORRESPONDING
ELEMENT STIFFNESS MATRIX AND CARRY OUT FURTHER ITER.

CHKGUS = .TRUE.
REJSTIFF = 1
CONVERG = 999.
GO TO 140
ENDIF

IF (EPS CRK.GT.0.0 .AND. EPS CRK.GE.EPSOLD) THEN

IF THE CRACK WAS UNLOADING (CLOSING) OR RELOADING-
(REOPENING) IN THE PREVIOUS ITERATION, THEN SET
THE STATE OF THE CRACK TO LOADING AGAIN.

IF (STATE.EQ.2 .OR. STATE.EQ.3.) CRACK(L, I CRACK + 117, J) = 1.0

THE CRACK IS STILL OPENING. SAVE THE CURRNET NORMAL
STRAIN AS THE CRACK'S OLD STRAIN.CRACK CLOSURE OR
UNLOADING IN SUBSEQUENT STEPS MAY START FROM THIS
STORED STRAIN.

\[ \text{CRACK}(L, 47+ICRACK, J) = \text{EPSCK} \]

SAVE THE CURRENT STRAIN FOR DISTINGUISING BETWEEN
UNLOADING OR RELOADING IN SUBSEQUENT STEPS.

\[ \text{CRACK}(L, 57+ICRACK, J) = \text{EPSCK} \]

SAVE THE SHEAR STRAIN ALONG THE CRACK.

\[ \text{CRACK}(L, 67+ICRACK, J) = \text{GMC1} \]

\[ \text{CRACK}(L, 77+ICRACK, J) = \text{GMC2} \]

IF THE CRACK WAS COMPLETELY OPEN ...

\[ \text{EINITL} = \text{PROPER(1)} \]

IF \((\text{DCRACK} \leq \text{EINITL}/1000.0)\) GO TO 140

DETERMINE THE SECANT 'E' CORRESPONDING TO THE
CURRENT CRACK STRAIN, EPSCK.

\[ \text{CALL CRACKE}(\text{EPSCK}, \text{PROPER}, \text{ES}, \text{ICRACK}, L, J) \]

IF \((\text{ES} \leq \text{EINITL}/1000.0)\) THEN

THE CRACK HAS COMPLETELY OPENED LEADING TO ZERO STIFF.
TO BE ABLE TO CALCULATE THE CRACK STRAIN IN SUBSEQUENT
ITERATIONS ARBITRARILY ASSIGN A SMALL 'E' FOR IT.

\[ \text{CRACK}(L, 87+ICRACK, J) = \text{EINITL}/1000.0 \]

\[ \text{CHKGUS} = \text{.TRUE.} \]

\[ \text{HSTIFF} = 1 \]

\[ \text{COHVER} = 999. \]

WRITE (*) 'CRKPHT 4 999 ELUMB', ELUMB
GO TO 140

ENDIF

CHECK FOR CONVERGENCE

\[ \text{FT} = \text{CRACK}(L, 37+ICRACK, J) \]

\[ \text{IFLAG} = 0 \ldots \text{CRACK STIFFNESS HAS CONVERGED} \]
\[ \text{IFLAG} = 1 \ldots \text{SHOULD BE UPDATED}. \]

\[ \text{IFLAG} = 0 \]

\[ \text{CALL CRKCON} \]

\[ \text{IF (SIGCRK} \leq \text{FT}) \text{GO TO 140} \]

\[ \text{.... CORRECTION FOR OVERSHOOTING} \]

\[ \text{IF (SIGCRK} \geq \text{FT} \leftarrow \text{AND. IFLAG EQ.0) THEN} \]

\[ \text{CALL OVERFT} \]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IF (CORDER .EQ. 999) WRITE (*, *)
    CRKPT 6 999 ELMNUM, ELMNUM
    GO TO 140
ENDIF

IF (EPSCKR .GT. 0.0 .AND. EPSCKR .LT. EPSOLD) THEN
    ...
    GET THE PREVIOUSLY REGISTERED CRACK STRAINS.
    PRECKR = CRACK(L,47+ICRACK,J)
   ...
    THE CRACK IS CLOSING. THE STATE OF THE CRACK SHOULD BE
    EQUAL TO 2.
    IF (EPSCKR .LT. PRECKR) CRACK(L,117+ICRACK,J) = 2.0
    ...
    THE CRACK IS RELOADING (OPENING). THE STATE OF THE CRACK
    MUST BE EQUAL TO 3.
    IF (EPSCKR .GE. PRECKR) CRACK(L,117+ICRACK,J) = 3.
    ...
    SAVE THE CURRENT CRACK STRAINS.
    CRACK(L,67+ICRACK,J) = EPSCKR
    CRACK(L,ICRACK+67,J) = GNCRR1
    CRACK(L,ICRACK+77,J) = GNCRR2
    GO TO 140
ENDIF

IF (EPSCKR .LE. 0.0) THEN
    ...
    COMPLETE CLOSURE OF THE CRACK
    ...

IF (EPSCKR .LE. 0.0) THEN
    ...
    IF THE CRACK CLOSES (**) THE STIFFNESS OF THE ELEMENT REFLECTS A
    DAMAGE PLANE IN THAT DIRECTION. IF THE CRACK REOPENS THEN IT STILL
    GOES BACK TO SECANT RELOADING STIFFNESS AT CLOSURE (**) OF THE CRACK.
    THE SHEAR DAMAGE AT THE INTERFACE IS ALLOWED TO CONTINUE AS BEFORE.
    IF (SENFG1 .EQ. 2) THEN
        CRACK(L,227+I,J) = CRACK(L,67+I,J)
        STATE=4.0
        ESINTO=HISTO(L,20,J)
        CRACK(L,67+I,J)=10.*ESINTC
        GO TO 30
END IF

IF A VERY STIFF CRACK IS CLOSING ELIMINATE THIS
CRACK FROM THE 'CRACK' ARRAY AND DO NOT CONSIDER
IT IN THE FOLLOWING ITERATIONS.

FOR THE LAST CRACK .......

EINITIAL = PROPER(1)

THIS IS CASE OF A WIDE OPEN CRACK DETECTED AS CLOSED! BECAUSE
THE STIFFNESS IS ARRESTED AT ECONC/1000. AND THE STRESS IS
NEGATIVE. THEREFORE NEGATIVE CRACK STRAINS!

IF (CRACK(L,87+ICRACK,J) .LE . EINITIAL/1000.0) GO TO 140
IF (ICRACK .EQ. HCRACK) THEN
CRACK(L,1,J) = CRACK(L,1,J) - 1.000
GO TO 160
ENDIF

CALL ARANGE (ICRACK, HCRACK, L, J)

140 CONTINUE
150 CONTINUE

INITIALIZE THE CONSTITUTIVE MATRIX FOR THE INTACT CONC.

IFG = 2
CALL PSTSTN (SIGMA, PVAL, PDIR, IFG, IOUT)
DO 180 JJ = 1, 6
DEP (JJ) = 0.0
170 CONTINUE
180 CONTINUE

RETRIEVE THE CONSTITUTIVE MATRIX (IN PRINCIPAL AXES)

ES = HISTO(L,20,J)
NUS = HISTO(L,21,J)
CONST = ES/((1.-2.0*NUS)*(1+NUS))
WRITE(17,818)ES,NUS
C818 FORMAT(1X,'ES=',D12.5,'NUS',D12.5)

DEP(1,1) = CONST*(1.0-NUS)
DEP(2,2) = CONST*(1.0-NUS)
DEP(3,3) = CONST* NUS
DEP(1,2) = DEP(1,2)
DEP(2,1) = DEP(1,2)
DEP(1,3) = DEP(1,2)
DEP(3,1) = DEP(1,2)
DEP(2,3) = DEP(1,2)
DEP(3,2) = DEP(1,2)
DEP(4,4) = ES/(2.0*(1+NUS))
DEP(5,5) = DEP(4,4)
DEP(6,6) = DEP(4,4)

INVERT THE 'DEP' MATRIX.

N = NO. OF ROWS IN 'DEP'

NRC = MAX. NO. OF ROWS AND COLUMNS ALLOWED.
EPS = MIN. ALLOWABLE MAGNITUDE FOR A PIVOT ELEMENT.
H  = 6  
HRC = 6  
EPS = 10.0E-20  

C DETER = DETERMINANT OF 'DEP' THAT IS RETURNED AS THE VALUE OF FUNCTION 'SIMUL'.  
C
DETER = SIMUL($DEP, EPS, HRC, IOUT)  
IF (DETER .EQ. 0.0) THEN
  WRITE (IOUT, 900) L, J, ELNUM
  STOP
ENDIF

C DETERMINE PRINCIPAL STRAINS CONTRIBUTED TO THE TOTAL STRAINS BY THE INTACT CONCRETE.
C
SIGMA1 = PVAL(1)  
SIGMA2 = PVAL(2)  
SIGMA3 = PVAL(3)  
SMAX = DMAX1(DMAX1(DABS(SIGMA1),DABS(SIGMA2)),DABS(SIGMA3))  
IF (DABS(SIGMA1) .LE. SMAX*1.E-4) SIGMA1 = 0.0  
IF (DABS(SIGMA2) .LE. SMAX*1.E-4) SIGMA2 = 0.0  
IF (DABS(SIGMA3) .LE. SMAX*1.E-4) SIGMA3 = 0.0  
SIGMA1 = DMIN1(PROPER(6),SIGMA1)  
EPS1 = DEP(1,1)*SIGMA1 + DEPC(1,2)*SIGMA2 + DEP(1,3)*SIGMA3  
EPS2 = DEP(2,1)*SIGMA1 + DEP(2,2)*SIGMA2 + DEP(2,3)*SIGMA3  
EPS3 = DEP(3,1)*SIGMA1 + DEP(3,2)*SIGMA2 + DEP(3,3)*SIGMA3  
RETURN

900 FORMAT(//,'ERROR','/,'NEGATIVE PIVOT ENCOUNTERED'  
  1 ,' DURING INVERSION OF CONSTITUTIVE MATRIX FOR INTACT '  
  2 ', ...CONCRETE ...CRACK STOP 3 ', 'G.PT', I2, 'LAYER', I2,  
  3 ', 'ELNUM', I2)  
END

C ---------------------------------------------------------------
C INCLUDE(PROCESS)  
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE  
C DRESNS SPGN
C ---------------------------------------------------------------
C
INTEGER  
COMBAR/BLOCK/IST0(27,70,16),CRACK(27,260,16),IST01,IBIST1  
$,IBIST2  
COMBAR/ABC/TRSF(27,60,16),TRESP,ITRGA,ITRGS2  
DIMENSION PROPER(28),DSTIF(6,6),TRESP(6,6),TEMPI(6,6),TEMPF(6,6),TEMPI1(6,6)  
REAL*8 CR(3,3),BSTL(3,3),ST(3,3),K(3,3)  
LOGICAL CHKGUS

DO 110 IP = 1, 6
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
DO 100 KP = 1, 6
DISTIF(IP,KP) = 0.0
100 CONTINUE

DO 130 IM = 1, 6
DO 120 IN = 1, 6
TEMP11(IM,IN) = 0.0
TEMP12(IM,IN) = 0.0
120 CONTINUE
130 CONTINUE
C
ICOUNT = 1
IFLAG1 = 0
IFLAG2 = 0
IFLAG3 = 0
IFLAG4 = 0
IFLAG5 = 0
IFLAG6 = 0
ICHECK = 0
MCHECK = 0
F3 = 0.0
F2 = 0.0
F1 = 0.0
ISPG2 = 0
ISPG3 = 0
C
DO 200 ISP = 1, 3
C
ISPG2 = 0
ISPG3 = 0
C
WRITE(*,*) 'FLAGS 1',IFLAG1,'FLAGS 2',IFLAG2,'FLAGS 3',IFLAG3
IF (TENSTF(L,ISP,J) .EQ. 1.0) THEN
MCHECK = 1
C
WRITE(*,*) 'MCHECK=1 ISP=',ISP,'ISPG',TENSTF(L,3+ISP,J)
IF (ISP .EQ. 1) IFLAG1 = 1
IF (ISP .EQ. 2) IFLAG2 = 1
IF (ISP .EQ. 3) IFLAG3 = 1
GO TO 190
ENDIF
C
CHECK THE STATUS OF PRESENT BUT INACTIVE TENSIONSTIFFENING SPRINGS
C
... THE CURRENT PRINCIPAL STRESS DIRECTIONS ARE--
C
N(1,1) = HISTO(L,10,J)
N(2,1) = HISTO(L,11,J)
N(3,1) = HISTO(L,12,J)
N(1,2) = HISTO(L,13,J)
N(2,2) = HISTO(L,14,J)
N(3,2) = HISTO(L,15,J)
N(1,3) = HISTO(L,16,J)
N(2,3) = HISTO(L,17,J)
N(3,3) = HISTO(L,18,J)
C
IF (TENSTF(L,ISP,J) .EQ. 0.0) THEN
ISPG = TENSTF(L,3+ISP,J)
C
IF (ISPG .EQ. 0) THEN
IF (ISP .EQ. 1) IFLAG1 = 1
IF (ISP .EQ. 2) IFLAG2 = 1
IF (ISP .EQ. 3) IFLAG3 = 1
GO TO 190
C RECOVER THE SPRING DIRECTION AND THE STRAINS IN THAT DIRECTION
C
ISPDCS = 9*(ISPG-1) + 1
AL1 = TESTFL(L,30+ISPDCS,J)
AM1 = TESTFL(L,31+ISPDCS,J)
AH1 = TESTFL(L,32+ISPDCS,J)
C STEEL DIRECTION STRAINS ARE . . . .
EPSSTL = EPSX(ALL+AL1*AL1 + EPSY(AM1*AM1 + EPSZ*AH1*AH1 +
GAMAX(ALL+AM1 + GAMAY(AM1 + GAMAZ*AH1*AH1 + GAMAIZ*AB1*AL1
IF (EPSSTL .GT. 0.0) THEN
TESTFL(L,69+ISPG,J) = EPSSTL
ELSE
TESTFL(L,69+ISPG,J) = 0.0
ENDIF
C CRACK STRAIN IS
EPSCR = TESTFL(L,9*ISPG,J)
WRITE(*,*) 'TSSTIF EPSSTL', EPSSTL, 'EPSCR ',EPSCR,'SPG ', ISPG
C
C CHECK IF ANY ACTIVE CRACK CONtributes TO ACTIVATING THE T.SPRING
C
BCRACK = INT(CRACK(L,1,J))
DO 180 ICK = 1, 3
IMG = 9*(ICK-1) + 1
NCR(1,1) = CRACK(L,137+IMG,J)
NCR(1,2) = CRACK(L,138+IMG,J)
NCR(1,3) = CRACK(L,139+IMG,J)
NCR(2,1) = CRACK(L,140+IMG,J)
NCR(2,2) = CRACK(L,141+IMG,J)
NCR(2,3) = CRACK(L,142+IMG,J)
NCR(3,1) = CRACK(L,143+IMG,J)
NCR(3,2) = CRACK(L,144+IMG,J)
NCR(3,3) = CRACK(L,145+IMG,J)
EPSCRK = CRACK(L,57+ICK,J)
EPSULT = CRACK(L,127+ICK,J)
EPSCKR = EPSULT-EPSCRK
EPSCKR = DMIN1(EPSCKR, EPSCKR)
SIGIS = CRACK(L,137+ICK,J)*(1.0-0.2*EPSCKR/EPSULT)
H(1,1) = TESTFL(L,180+ISPG,J)
H(1,2) = TESTFL(L,181+ISPG,J)
H(1,3) = TESTFL(L,182+ISPG,J)
H(2,1) = TESTFL(L,183+ISPG,J)
H(2,2) = TESTFL(L,184+ISPG,J)
H(2,3) = TESTFL(L,185+ISPG,J)
H(3,1) = TESTFL(L,186+ISPG,J)
H(3,2) = TESTFL(L,187+ISPG,J)
H(3,3) = TESTFL(L,188+ISPG,J)
DO 150 IMM = 1, 3
DO 140 IMP = 1, 3
BSTL(IMM,IMP) = 0.0
BSTL(IMM,IMP) = BSTL(IMM,IMP) + H(IMM,1)*NCR(1,IMP)
BSTL(IMM,IMP) = BSTL(IMM,IMP) + H(IMM,2)*NCR(1,IMP)
BSTL(IMM,IMP) = BSTL(IMM,IMP) + H(IMM,3)*NCR(1,IMP)
CONTINUE
180 CONTINUE
150 CONTINUE
DO KJ = 1, 3
DO KR = 1, 3
IF (DSUBS(BSTL(KJ,KR)) .LE. 0.01) BSTL(KJ,KR) = 0.0
1
ESD DO
END DO
AL1 = BSTL(1,1)
AM1 = BSTL(2,1)
AN1 = BSTL(3,1)
AL2 = BSTL(1,2)
AM2 = BSTL(2,2)
AN2 = BSTL(3,2)
AL3 = BSTL(1,3)
AM3 = BSTL(2,3)
AN3 = BSTL(3,3)
DOT=SCR(1,1)*N(1,1)+SCR(1,2)*N(1,2)+SCR(1,3)*N(1,3)
IF (DABS(DOT) .GE. 0.250) THEN
C CHECK FOR SPECIAL CASES......
C
IF (TESTFL(1,4,J).EQ.0 .OR. TESTFL(1,5,J).EQ.0 .OR. TESTFL(1,6,J).EQ.0 .OR. TESTFL(2,4,J).EQ.0 .OR. TESTFL(2,5,J).EQ.0 .OR. TESTFL(2,6,J).EQ.0 .OR. TESTFL(3,4,J).EQ.0 .OR. TESTFL(3,5,J).EQ.0 .OR. TESTFL(3,6,J).EQ.0) THEN
AL1 = BSTL(1,1)
AM1 = BSTL(2,1)
AN1 = BSTL(3,1)
AL2 = BSTL(1,2)
AM2 = BSTL(2,2)
AN2 = BSTL(3,2)
AL3 = BSTL(1,3)
AM3 = BSTL(2,3)
AN3 = BSTL(3,3)
ENDIF
ENDIF
C
IF (TESTFL(1,4,J).EQ.0 .OR. TESTFL(1,5,J).EQ.0 .OR. TESTFL(1,6,J).EQ.0 .OR. TESTFL(2,4,J).EQ.0 .OR. TESTFL(2,5,J).EQ.0 .OR. TESTFL(2,6,J).EQ.0 .OR. TESTFL(3,4,J).EQ.0 .OR. TESTFL(3,5,J).EQ.0 .OR. TESTFL(3,6,J).EQ.0) THEN
AL1 = BSTL(1,1)
AL2 = BSTL(1,3)
AL3 = BSTL(1,2)
AM1 = BSTL(2,1)
AM2 = BSTL(2,3)
AM3 = BSTL(2,2)
AN1 = BSTL(3,1)
AN2 = BSTL(3,3)
AN3 = BSTL(3,2)
ENDIF
ENDIF
C
SIG1HH = 0.0
IF (TESTFL(1,4,J).GT.0.0 .AND. TESTFL(1,5,J).GT.0.0 .AND. TESTFL(1,6,J).GT.0.0 .AND. TESTFL(2,4,J).GT.0.0 .AND. TESTFL(2,5,J).GT.0.0 .AND. TESTFL(2,6,J).GT.0.0 .AND. TESTFL(3,4,J).GT.0.0 .AND. TESTFL(3,5,J).GT.0.0 .AND. TESTFL(3,6,J).GT.0.0) THEN
SIG1HH = SIG1HH
ELSE IF (TESTFL(1,4,J).GT.0.0 .AND. TESTFL(1,5,J).GT.0.0 .AND. TESTFL(1,6,J).GT.0.0 .AND. TESTFL(2,4,J).GT.0.0 .AND. TESTFL(2,5,J).GT.0.0 .AND. TESTFL(2,6,J).GT.0.0 .AND. TESTFL(3,4,J).GT.0.0 .AND. TESTFL(3,5,J).GT.0.0 .AND. TESTFL(3,6,J).GT.0.0) THEN
SIG1HH = SIG1HH*((AL3+AM3+AN3)*(AM3+AH3-AM3)
*AM2)-(AM1+AM3-AM1+AH3)+(AL2+AM3-AM2+AL3)
)/(AM3+AM3+AM3+AL2+AL3+AL3+AL3)
ELSE IF (TESTFL(1,4,J).GT.0.0 .AND. TESTFL(1,5,J)
323

323

1 .EQ. 0.0 .AND. TESTSF(L,6,J).EQ.0.0 .OR.
2 TESTSF(L,6,J).GT.0.0 .AND. TESTSF(L,6,J)
3 .EQ.0.0 .AND. TESTSF(L,4,J).EQ.0.0 .OR.
4 TESTSF(L,4,J).GT.0.0 .AND. TESTSF(L,6,J)
5 .EQ.0.0 .AND. TESTSF(L,9,J).EQ.0.0 THEN
6 SIG1HN = SIG1H+((AM1+AM2-AM1-AM2)+(AL3+AM2-AM3)
7 *(AL2)-(AL2+AM1-AM2+AL1)+(AM3*AM2-AM3+AM2))
8 /((AL1+AM2*(AM3*AM2-AM3+AM2))

ENDIF
IF (TESTSF(L,ISP,J) .EQ. 0.0) THEN
IF (EPSSTL .GE. EPSCR) THEN
IF (ISP .EQ. 1) ICK1 = ICK
2 IF (ISP .EQ. 2) ICK2 = ICK
3 IF (ISP .EQ. 3) ICK3 = ICK
4 ICHECK = 1
5 ICOUP = 0
6 MCHECK = 0
7 IF (ISP .EQ. 1) IFLAG1 = 1
8 IF (ISP .EQ. 2) IFLAG2 = 1
9 IF (ISP .EQ. 3) IFLAG3 = 1
10 TESTSF(L,ISP,J) = 1.0
11 TESTSF(L,6+ISPG,J) = SIG1HN
12 TEHSF(L,15+ISPG,J) = EPSCR
13 TESTSF(L,16+ISPG,J) = EPSCR
14 TESTSF(L,18+ISPG,J) = EPSCR
15 EPSCR
16 ELSE IF (ISOFST .EQ. 1) THEN
17 MCHECK = 0
18 ICOUP = 0
19 IF (ISP .EQ. 1) IFLAG4 = 2
20 IF (ISP .EQ. 2) IFLAG5 = 2
21 IF (ISP .EQ. 3) IFLAG6 = 2
22 TESTSF(L,6+ISPG,J) = SIG1HN
23 TESTSF(L,15+ISPG,J) = EPSSTL
24 TESTSF(L,16+ISPG,J) = EPSSTL
25 EPSSTL
26 IF (EPSSTL .GT. 0.0) TESTSF(L,24+ISPG,J)
27 = SIG1HN/EPSSTL
28 ELSE IF (EPSSTL.GT.0.0 .AND. EPSSTL.LT.EPSCR)
29 THEN
30 MCHECK = 0
31 ICOUP = 0
32 IF (ISP .EQ. 1) IFLAG4 = 2
33 IF (ISP .EQ. 2) IFLAG5 = 2
34 IF (ISP .EQ. 3) IFLAG6 = 2
35 TESTSF(L,6+ISPG,J) = SIG1HN
36 TESTSF(L,15+ISPG,J) = EPSSTL
37 TESTSF(L,16+ISPG,J) = EPSSTL
38 EPSSTL
39 IF (EPSSTL .GT. 0.0) TESTSF(L,24+ISPG,J)
40 = SIG1HN/EPSSTL
41 ENDIF
42 ENDIF
43 ENDIF
44 CONTINUE
45 ENDIF
46 CONTINUE
47 CONTINUE
48 C ... IF THE CURRENT CRACK IS NOT STRAIN SOFTENING THEN
49 C
50 ECRAK = INT(CRACK(L,1,J))

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IF (MCHECK .EQ. 0) THEN
  HCRACK = IHT(CRACK(L,1,J))
  IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.1) THEN
    ITCRK = HCRACK
  ELSE IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.0 .OR.
    1 IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.0 .OR.
    2 IFLAG2.EQ.1 .AND. IFLAG3.EQ.0 .OR. IFLAG3.EQ.1) THEN
    IF (IFLAG4.EQ.2 .OR. IFLAG6.EQ.2 .OR. IFLAG6.EQ.2) THEN
      IF (TESTF(L,1,.J).EQ.1 .AND. IFLAG1.EQ.1 .OR. TESTF(L,
        2,.J).EQ.1 .AND. IFLAG2.EQ.1 .OR. TESTF(L,3,.J))
        1 .EQ.1 .AND. IFLAG3.EQ.1) THEN
        IF (IFLAG4.EQ.2) TESTF(L,1,.J) = 1.0
        IF (IFLAG6.EQ.2) TESTF(L,2,.J) = 1.0
        IF (IFLAG6.EQ.2) TESTF(L,3,.J) = 1.0
        ITCRK = HCRACK
        IFLAG1 = 1
        IFLAG2 = 1
        IFLAG3 = 1
      ENDIF
      WRITE (*,*) 'ALL FLAGS ON CASE A'
    ELSE IF (TESTF(L,4,.J).EQ.0 .OR. TESTF(L,5,.J).EQ.0 .OR.
      1 .OR. TESTF(L,6,.J).EQ.0) THEN
      ITCRK = HCRACK
      IFLAG1 = 1
      IFLAG2 = 1
      IFLAG3 = 1
      WRITE (*,*) 'ALL FLAGS ON CASE B'
    ENDIF
  ELSE IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.0 .AND. IFLAG3.EQ.0 .OR.
    1 IFLAG1.EQ.1 .AND. IFLAG3.EQ.0 .AND. IFLAG2.EQ.0 .OR.
    2 IFLAG2.EQ.1 .AND. IFLAG3.EQ.0 .AND. IFLAG1.EQ.0 .OR.
    2 IFLAG3.EQ.1 .AND. IFLAG1.EQ.0 .AND. IFLAG2.EQ.0) THEN
    IF (IFLAG4.EQ.2 .AND. IFLAG6.EQ.2 .AND. IFLAG6.EQ.2 .OR.
      1 IFLAG4.EQ.2 .AND. IFLAG6.EQ.2 .AND. IFLAG6.EQ.2 .OR.
      2 IFLAG6.EQ.2 .AND. IFLAG4.EQ.2 .AND. IFLAG4.EQ.2 .OR.
      2 IFLAG6.EQ.2 .AND. IFLAG6.EQ.2 .AND. IFLAG6.EQ.2) THEN
      IF (IFLAG4.EQ.1 .AND. TESTF(L,6,.J).GT.0.0 .OR. IFLAG6.EQ.2)
        1 .EQ.1 .AND. TESTF(L,5,.J).GT.0.0 .OR. IFLAG1.EQ.1)
        2 .AND. TESTF(L,6,.J).GT.0.0) THEN
        ITCRK = HCRACK
        IF (IFLAG4.EQ.2) TESTF(L,1,.J) = 1.0
        IF (IFLAG6.EQ.2) TESTF(L,2,.J) = 1.0
        IF (IFLAG6.EQ.2) TESTF(L,3,.J) = 1.0
        IFLAG1 = 1
        IFLAG2 = 1
        IFLAG3 = 1
      ENDIF
      WRITE (*,*) 'ALL FLAGS ON CASE C'
    ELSE IF (TESTF(L,4,.J).EQ.1.0 .AND. TESTF(L,5,.J).EQ.0.0 .OR.
      1 .AND. TESTF(L,6,.J).EQ.0.0 .OR. TESTF(L,5,.J).EQ.0.0 .OR.
      2 .AND. TESTF(L,4,.J).EQ.1.0 .AND. TESTF(L,6,.J).EQ.0.0 .OR.
      3 .AND. TESTF(L,4,.J).EQ.1.0 .AND. TESTF(L,6,.J).EQ.0.0 .OR.
      4 .AND. TESTF(L,4,.J).EQ.1.0) THEN
        ITCRK = HCRACK
        IFLAG1 = 1
        IFLAG2 = 1
        IFLAG3 = 1
      ENDIF
    ENDIF
  ENDIF
ENDIF
ENDIF
IF (IFLAG1.EQ.1 .AND. IFLAG2.EQ.1 .AND. IFLAG3.EQ.1) THEN
  ICHECK = 999.0
  WRITE (*,*) 'TSTIF 1 999 ELEUM', ELEUM
  BSTIFF = 1
ENDIF
CHEGS = .TRUE.
$$IMG = 0$$

**DO** 290 **ISP** = 1, 3

**ISPG** = **TNSTF**(*)*,ISP, J*)

**DO** 220 **IM** = 1, 6

**DO** 210 **IM** = 1, 6

**D** **E** **N** **Y** *(IM, IMB) = 0.0

**210** **C** **O** **N** **T** **I** **U** **N** **E**

**220** **C** **O** **N** **T** **I** **U** **N** **E**

**IF** (**ISP.GT.0** .A**N**D. **TNSTF**(*)*,ISP, J*) .**E** **Q** .1.0) **T** **H** **E** **N**

**IMG** = (**ISP.GT.1**)*0 + 1

**C**

**C** . . . **T** **E** **N** **S** **I** **S** **I** **O** **N** ** (S** **T** **E** **E** **L) **D** **I** **R** **E** **C** **T** **I** **O** **N** W. R. T **G** **L** **O** **B** **R** **L** **A** **L**

**C**

$$H(1,1) = TNSTF(L, LMG = 30, J)$$

$$H(1,2) = TNSTF(L, LMG = 31, J)$$

$$H(1,3) = TNSTF(L, LMG = 32, J)$$

$$H(2,1) = TNSTF(L, LMG = 33, J)$$

$$H(2,2) = TNSTF(L, LMG = 34, J)$$

$$H(2,3) = TNSTF(L, LMG = 35, J)$$

$$H(3,1) = TNSTF(L, LMG = 36, J)$$

$$H(3,2) = TNSTF(L, LMG = 37, J)$$

$$H(3,3) = TNSTF(L, LMG = 38, J)$$

$$EPSTL1 = EPSI*H(1,1)*H(1,1) + EPSY*H(1,2)*H(1,2) + EPSZ*H(1,3)*H(1,3)$$

$$1 + EPSZ*H(2,1)*H(2,1) + EPSY*H(2,2)*H(2,2) + EPSZ*H(2,3)*H(2,3)$$

$$2 + EPSZ*H(3,1)*H(3,1) + EPSY*H(3,2)*H(3,2) + EPSZ*H(3,3)*H(3,3)$$

$$EPSTL2 = EPSI*H(1,1)*H(1,1) + EPSY*H(1,2)*H(1,2) + EPSZ*H(1,3)*H(1,3)$$

$$1 + EPSZ*H(2,1)*H(2,1) + EPSY*H(2,2)*H(2,2) + EPSZ*H(2,3)*H(2,3)$$

$$2 + EPSZ*H(3,1)*H(3,1) + EPSY*H(3,2)*H(3,2) + EPSZ*H(3,3)*H(3,3)$$

$$EPSTL3 = EPSI*H(1,1)*H(1,1) + EPSY*H(1,2)*H(1,2) + EPSZ*H(1,3)*H(1,3)$$

$$1 + EPSZ*H(2,1)*H(2,1) + EPSY*H(2,2)*H(2,2) + EPSZ*H(2,3)*H(2,3)$$

$$2 + EPSZ*H(3,1)*H(3,1) + EPSY*H(3,2)*H(3,2) + EPSZ*H(3,3)*H(3,3)$$

**C**

$$HCR(1,1) = HISTO(L, LM1, J)$$

$$HCR(1,2) = HISTO(L, LM2, J)$$

$$HCR(1,3) = HISTO(L, LM3, J)$$

$$HCR(2,1) = HISTO(L, LM4, J)$$

$$HCR(2,2) = HISTO(L, LM5, J)$$

$$HCR(2,3) = HISTO(L, LM6, J)$$

$$HCR(3,1) = HISTO(L, LM7, J)$$

$$HCR(3,2) = HISTO(L, LM8, J)$$

$$HCR(3,3) = HISTO(L, LM9, J)$$

**DO** 240 **IMM** = 1, 3

**DO** 230 **IMP** = 1, 3

**B** **S** **T** **L** (**IMM, IMM** = 0.0

**B** **S** **T** **L** (**IMM, IMM** = **B** **S** **T** **L** (**IMM, IMM** + (**H** (**IMM, 1**) + **B** (**IMP, 1**)

**B** **S** **T** **L** (**IMM, IMM** = **B** **S** **T** **L** (**IMM, IMM** + (**H** (**IMM, 2**) + **B** (**IMP, 2**)

**B** **S** **T** **L** (**IMM, IMM** = **B** **S** **T** **L** (**IMM, IMM** + (**H** (**IMM, 3**) + **B** (**IMP, 3**)

**220** **C** **O** **N** **T** **I** **U** **N** **E**

**230** **C** **O** **N** **T** **I** **U** **N** **E**

**240** **C** **O** **N** **T** **I** **U** **N** **E**

**DO** **MN** = 1, 3

**DO** **MJ** = 1, 3

**I** **F** (**ABS** (**B** **S** **T** **L** (**MN, MJ**)) .**L** **E**. 0.01) **B** **S** **T** **L** (**MN, MJ** = 0.0

**E** **N** **D** **D**

**A** **L** **L** **1** = **B** **S** **T** **L** (**1, 1**

**A** **L** **1** = **B** **S** **T** **L** (**1, 2**

**A** **L** **2** = **B** **S** **T** **L** (**1, 3**

**A** **L** **2** = **B** **S** **T** **L** (**2, 1**

**A** **L** **2** = **B** **S** **T** **L** (**2, 2**

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ AD2 = BSTL(2,3) \]
\[ AL3 = BSTL(3,1) \]
\[ AM3 = BSTL(3,2) \]
\[ AN3 = BSTL(3,3) \]
\[ PEPMAX = HISTO(L,41,J) \]
\[ PEPMD = HISTO(L,42,J) \]
\[ PEPMIN = HISTO(L,43,J) \]
\[ PSL1 = PEPMAX\times AL1 + PEPMD\times AM1 + PEPMIN\times AN1 \]
\[ PSL2 = PEPMAX\times AL2 + PEPMD\times AM2 + PEPMIN\times AN2 \]
\[ PSL3 = PEPMAX\times AL3 + PEPMD\times AM3 + PEPMIN\times AN3 \]
\[ TEDXYZ(1,1) = TEDSTF(L,24\times ISP\times J) \]
\[ IMG = IMG + 1 \]
\[ IF (IMG \times EQ. 1) THEN \]
\[ IF (ISP \times EQ. 1) THEN \]
\[ ISP1 = ISP \]
\[ ISP2 = 2 \]
\[ ISP3 = 3 \]
\[ ELSE IF (ISP \times EQ. 2) THEN \]
\[ ISP1 = ISP \]
\[ ISP2 = 3 \]
\[ ISP3 = 1 \]
\[ ELSE IF (ISP \times EQ. 3) THEN \]
\[ ISP1 = ISP \]
\[ ISP2 = 1 \]
\[ ISP3 = 2 \]
\[ ENDIF \]
\[ ISP2 = 0 \]
\[ ISP3 = 0 \]
\[ IF (TEST(L,4,J) \times EQ. ISP02) ISP2 = 1 \]
\[ IF (TEST(L,5,J) \times EQ. ISP02) ISP2 = 2 \]
\[ IF (TEST(L,6,J) \times EQ. ISP02) ISP2 = 3 \]
\[ IF (TEST(L,4,J) \times EQ. ISP03) ISP3 = 1 \]
\[ IF (TEST(L,5,J) \times EQ. ISP03) ISP3 = 2 \]
\[ IF (TEST(L,6,J) \times EQ. ISP03) ISP3 = 3 \]
\[ CHST12 = 0.0 \]
\[ CHST23 = 0.0 \]
\[ CDST13 = 0.0 \]
\[ IF (TEST(L,ISP3,J) \times EQ. 0) THEN \]
\[ IF (PSTL2 \times LE. TEST(L,9\times ISP\times J) \times LD.) \]
\[ EPSL2 = EPSL2 \times GT. TEST(L,9\times ISP\times J) \] THEN \]
\[ CHKUS = .TRUE. \]
\[ CONVER = 999.0 \]
\[ WRITE (+, *) 'TSTIF 2 999 ELEUM', \]
\[ ELEUM \]
\[ ENDIF \]
\[ ENDIF \]
\[ IF (TEST(L,ISP3,J) \times EQ. 0) THEN \]
\[ IF (PSTL3 \times LE. TEST(L,9\times ISP\times J) \times LD.) \]
\[ EPSL3 = EPSL3 \times GT. TEST(L,9\times ISP\times J) \] THEN \]
\[ CHKUS = .TRUE. \]
\[ CONVER = 999.0 \]
\[ WRITE (+, *) 'TSTIF 3 999 ELEUM', \]
\[ ELEUM \]
\[ ENDIF \]
\[ ENDIF \]
IF (TESTFL(ISP1,J).EQ.1.0 .AND. TESTFL(ISP2,J).EQ.1.0) THEN
  CSTN12 = DSQRT ((TESTFL(ISP1,J)+TESTFL(ISP2,J))
           /HISTO(L,26,J))
ELSE IF (TESTFL(ISP1,J).EQ.1.0 .AND. TESTFL(ISP2,J).EQ.0.0) THEN
  CSTN12 = DSQRT ((TESTFL(ISP1,J)+CSTN3)
                   /HISTO(L,26,J))
  IF (ESTPL1.GT. TESTFL(L,9+ISP1,J))
    CSTN12 = 0.0
ELSIF (TESTFL(ISP1,J).EQ.0.0 .AND. TESTFL(ISP2,J).EQ.1.0) THEN
  CSTN12 = DSQRTCSTN12
  IF (ESTPL1.GT. TESTFL(L,9+ISP1,J))
    CSTN12 = 0.0
ELSE IF (TESTFL(ISP1,J).EQ.0.0 .AND. TESTFL(ISP2,J).EQ.0.0) THEN
  CSTN12 = DSQRTCSTN12
  IF (ESTPL1.GT. TESTFL(L,9+ISP1,J))
    CSTN12 = 0.0
ENDIF

IF (TESTFL(ISP1,J).EQ.1.0 .AND. TESTFL(ISP3,J).EQ.1.0) THEN
  CSTN13 = DSQRT ((TESTFL(ISP1,J)+TESTFL(ISP3,J))
                   /HISTO(L,26,J))
ELSE IF (TESTFL(ISP1,J).EQ.1.0 .AND. TESTFL(ISP3,J).EQ.0.0) THEN
  CSTN13 = DSQRT ((TESTFL(ISP1,J)+CSTN3)
                   /HISTO(L,26,J))
  IF (ESTPL1.GT. TESTFL(L,9+ISP1,J))
    CSTN13 = 0.0
ELSIF (TESTFL(ISP1,J).EQ.0.0 .AND. TESTFL(ISP3,J).EQ.1.0) THEN
  CSTN13 = DSQRT ((TESTFL(ISP1,J)+CSTN3)
                   /HISTO(L,26,J))
  IF (ESTPL1.GT. TESTFL(L,9+ISP1,J))
    CSTN13 = 0.0
ELSIF (TESTFL(ISP1,J).EQ.0.0 .AND. TESTFL(ISP3,J).EQ.0.0) THEN
  CSTN13 = DSQRTCSTN13
  IF (ESTPL1.GT. TESTFL(L,9+ISP1,J))
    CSTN13 = 0.0
ENDIF

IF (TESTFL(ISP2,J).EQ.1.0 .AND. TESTFL(ISP3,J).EQ.1.0) THEN
  CSTN23 = DSQRT ((TESTFL(ISP2,J)+TESTFL(ISP3,J))
                   /HISTO(L,26,J))
ELSE IF (TESTFL(ISP2,J).EQ.1.0 .AND. TESTFL(ISP3,J).EQ.0.0) THEN
  CSTN23 = DSQRT ((TESTFL(ISP2,J)+CSTN3)
                   /HISTO(L,26,J))
  IF (ESTPL1.GT. TESTFL(L,9+ISP1,J))
    CSTN23 = 0.0
ELSIF (TESTFL(ISP2,J).EQ.0.0 .AND. TESTFL(ISP3,J).EQ.1.0) THEN
  CSTN23 = DSQRT ((TESTFL(ISP2,J)+CSTN3)
                   /HISTO(L,26,J))
  IF (ESTPL1.GT. TESTFL(L,9+ISP1,J))
    CSTN23 = 0.0
ELSIF (TESTFL(ISP2,J).EQ.0.0 .AND. TESTFL(ISP3,J).EQ.0.0) THEN
  CSTN23 = DSQRTCSTN3
  IF (ESTPL1.GT. TESTFL(L,9+ISP1,J))
    CSTN23 = 0.0
ENDIF

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


```
1 HISTO(L,26,J)*HISTO(L,20,J)/HISTO(L,26,J))
2 IF (EPSLT2 .GT. TEHSTFCL,9+ISPG,J))
1 CBST23 = 0.0
2 ELSE IF (TEHSTFCL,ISP3,J).El}.0.0 .AHD.
3 TEHSTFCL,ISP2,J).EQ.1.0) THEH
4 CHST23 = DSQRT CTEHSTFCL ,24+ISPG2,  J )  /
5 HISTOCL,26,J)*HISTGCL,20,J)/EISTOC
6 IF CEPSTL3 .GT. TEHSTFCL,9+ISPG, J) )
7 CHST23 =0.0
8 ELSE IF CTEHSTFCL,ISP2,J).EQ.0.0 .ABD.
9 TEHSTFCL,ISP3.J).EQ.0.0) THEH
10 CHST23 = DSQRT CHISTO CL,  20, J) /HISTO CL,
11 HISTOCL,26,J)+HISTOCL,20,J)/HISTOCL,26,J))
12 IF CEPSTL2.GT.TEHSTFCL,9+ISPG,J) .OR.
13 EPSTL3.GT.TEHSTFCL(L,9+ISPG,J))
14 CHST23 =0.0
15 EDDIF
16 TEMPllC2,l) = CHST12*HISTOCL,26,J)*HISTOCL
17 ,27,J)
18 TEHP11(1,2) = CHST12*HIST0CL,26,J)*HIST0CL
19 ,27,J)
20 TEMPllC3,l) = CHST13*HISTOCL,26,J)*HISTOCL
21 ,27,J)
22 TEMPll Cl, 3) = CHST13*HIST0CL,26,J)*HIST0CL
23 ,27,J)
24 TEHPllC2,3) = CDST23*HIST0CL,26,J)*HIST0CL
25 ,27,J)
26 TEMPllC3,2) = CHST23+EIST0 CL, 26, J) +HISTO CL
27 ,27,J)
28 CALL DTRAHS (TEPPII, H, TOUT)
29 ICOUP = 1
30 EDDIF
31 EDDIF
32 C
33 CALL DTRAHS (TENXYZ, H, IOUT)
34 EDDIF
35 DO 280 IJ = 1, 6
36 DO 270 JK = 1, 6
37 DTSTIFCIJ.JK) = TEHXYZCIJ.JK) + DTSTIFCIJ.JK)
38 CONTINUE
39 270 CONTINUE
40 280 CONTINUE
41 EDDIF
42 EDDIF
43 IF (ICHECK .EQ. 1) RETURN
44 IMG = 0
45 ICOUP = 1
46 DO 420 ISP = 1, 3
47 DO 310 IP = 1, 6
48 DO 300 KP = 1, 6
49 TERNXYZ(IP,KP) = 0.0
50 CONTINUE
51 300 CONTINUE
52 310 CONTINUE
53 C
54 C ... THE CURRENT PRINCIPAL STRESS DIRECTIONS ARE--
55 C
56 N(1,1) = HISTO(L,10,J)
57 N(2,1) = HISTO(L,11,J)
58 N(3,1) = HISTO(L,12,J)
59 N(1,2) = HISTO(L,13,J)
60 N(2,2) = HISTO(L,14,J)
```
\[ H(3,2) = \text{HIST0}(L,15,J) \]
\[ H(1,3) = \text{HIST0}(L,16,J) \]
\[ H(2,3) = \text{HIST0}(L,17,J) \]
\[ H(3,3) = \text{HIST0}(L,18,J) \]

IF (TESTFL(L,ISP,J) .EQ. 1.0) THEN

\[ \text{IFLAG1} = 1 \]
\[ \text{IFLAG2} = 1 \]
\[ \text{IFLAG3} = 1 \]
\[ \text{ISPG} = \text{TESTFL}(L,3*ISP,J) \]
\[ \text{ISPCDS} = 9*(ISPG-1) + 1 \]
\[ \text{AL1} = \text{TESTFL}(L,30*ISPCDS,J) \]
\[ \text{AM1} = \text{TESTFL}(L,31*ISPCDS,J) \]
\[ \text{AN1} = \text{TESTFL}(L,32*ISPCDS,J) \]
\[ \text{EPSSTL} = \text{EPSX} \times \text{AL1} \times \text{AL1} + \text{EPSY} \times \text{AM1} \times \text{AM1} + \text{EPSZ} \times \text{AN1} \times \text{AN1} + \text{GAMAXY} \times \text{AL1} \times \text{AM1} + \text{GAMAZY} \times \text{AM1} \times \text{AN1} + \text{GAMAZX} \times \text{AN1} \times \text{AL1} \]

\[ \text{IMHG} = (ISPG-1)*9 + 1 \]
\[ \text{NT}(1,1) = \text{TESTFL}(L,30*IMHG,J) \]
\[ \text{NT}(1,2) = \text{TESTFL}(L,31*IMHG,J) \]
\[ \text{NT}(1,3) = \text{TESTFL}(L,32*IMHG,J) \]
\[ \text{NT}(2,1) = \text{TESTFL}(L,33*IMHG,J) \]
\[ \text{NT}(2,2) = \text{TESTFL}(L,34*IMHG,J) \]
\[ \text{NT}(2,3) = \text{TESTFL}(L,35*IMHG,J) \]
\[ \text{NT}(3,1) = \text{TESTFL}(L,36*IMHG,J) \]
\[ \text{NT}(3,2) = \text{TESTFL}(L,37*IMHG,J) \]
\[ \text{NT}(3,3) = \text{TESTFL}(L,38*IMHG,J) \]

DO 320 IM = 1, 3

DO 320 IP = 1, 3

\[ \text{BSTL}(IM,IP) = 0.0 \]
\[ \text{BSTL}(IM,IP) = \text{BSTL}(IM,IP) + \text{BT}(1,1) \times \text{BT}(1,IP) \]
\[ \text{BSTL}(IM,IP) = \text{BSTL}(IM,IP) + \text{BT}(1,2) \times \text{BT}(2,IP) \]
\[ \text{BSTL}(IM,IP) = \text{BSTL}(IM,IP) + \text{BT}(1,3) \times \text{BT}(3,IP) \]

CONTINUE

320 CONTINUE

DO IR = 1, 3

DO HR = 1, 3

IF (DABS(BSTL(IR,HR)) .LE. 0.01) BSTL(IR,HR) = 0.0

END DO

END DO

AL2 = BSTL(2,1)
AM2 = BSTL(2,2)
AN2 = BSTL(2,3)
AL3 = BSTL(3,1)
AM3 = BSTL(3,2)

AN3 = BSTL(3,3)

EPSCOR = 0.0

EPSYL = TESTFL(L,12*ISP,G,J)

IF (EPSYL .LE. EPSYL) THEN

BRCAC = INT(TESTFL(L,ISP,J))

CALL SPGH (EPSYL, L, J, TOLER, CHKUS, PROPER, BSTIFF, CVER, IOUT, BRCAC, ISP, ELNUM, BSTIFF)

ELSE

TESTFL(L,ISP+24,J) = 0.10*TESTFL(L,ISP+6,J)/EPSYL

IF (TESTFL(L,18*ISP,J).LT.EPSYL .AND. EPSYL.GE. EPSYL) THEN

CHKUS = .TRUE.

BSTIFF = 1

CVER = 9999.0

WRITE (*, •) 'TESTFL 4 999 ELNUM', ELNUM

ENDIF

TESTFL(L,15*ISP,J) = EPSYL

TESTFL(L,18*ISP,J) = EPSYL

ENDIF


```fortran
TEHSY$1(1,1) = TEHSY$L(1,4+ISPG, J)
IMG = IMG + 1
IF (IMG .EQ. 1) THEN
  IF (ICOUP .EQ. 1) THEN
    ISP01 = ISP02 = ISP03 = 2
    ELSE IF (ISPG .EQ. 2) THEN
      ISP02 = 2
      ISP03 = 1
    ELSE IF (ISPG .EQ. 3) THEN
      ISP01 = ISP03 = 1
      ISP02 = 2
    ENDIF
  ENDIF
  IF (TEHSY$2(L,4,J) .EQ. ISPG2) ISP2 = 1
  IF (TEHSY$2(L,5,J) .EQ. ISPG2) ISP2 = 2
  IF (TEHSY$2(L,6,J) .EQ. ISPG2) ISP2 = 3
  ELSE IF (ISPG .EQ. 2) THEN
    ISP01 = ISP03 = 1
    ISP02 = 3
  ELSE IF (ISPG .EQ. 3) THEN
    ISP01 = ISP03 = 3
    ISP02 = 1
  ENDIF
  EHDIF
  IF (TEHSY$2(L,4,J) .EQ. ISPG3) ISP3 = 1
  IF (TEHSY$2(L,5,J) .EQ. ISPG3) ISP3 = 2
  IF (TEHSY$2(L,6,J) .EQ. ISPG3) ISP3 = 3
  EPSTL1 = EPSX*BT(1,1)*BT(1,1) + EPSY*BT(1,2)*BT(1,2) + EPSZ*BT(1,3)*BT(1,3) + GAMAX*BT(1,1)*BT(1,2) + GAMAY*BT(1,2)*BT(1,3) + GAMAZ*BT(1,3)*BT(1,1)
  EPSTL2 = EPSX*BT(2,1)*BT(2,1) + EPSY*BT(2,2)*BT(2,2) + EPSZ*BT(2,3)*BT(2,3) + GAMAX*BT(2,1)*BT(2,2) + GAMAY*BT(2,2)*BT(2,3) + GAMAZ*BT(2,3)*BT(2,1)
  EPSTL3 = EPSX*BT(3,1)*BT(3,1) + EPSY*BT(3,2)*BT(3,2) + EPSZ*BT(3,3)*BT(3,3) + GAMAX*BT(3,1)*BT(3,2) + GAMAY*BT(3,2)*BT(3,3) + GAMAZ*BT(3,3)*BT(3,1)
  HCR(1,1) = HIST0(L,44,J)
  HCR(1,2) = HIST0(L,45,J)
  HCR(1,3) = HIST0(L,46,J)
  HCR(2,1) = HIST0(L,47,J)
  HCR(2,2) = HIST0(L,48,J)
  HCR(2,3) = HIST0(L,49,J)
  HCR(3,1) = HIST0(L,50,J)
  HCR(3,2) = HIST0(L,51,J)
  HCR(3,3) = HIST0(L,52,J)
  DO 370 IMM = 1, 3
    DO 360 IMP = 1, 3
      BSTL(IMM,IMP) = 0.0
      BSTL(IMM,IMP) = BSTL(IMM,IMP) + BT(IMM,1)*ECR(IMP,1)
      BSTL(IMM,IMP) = BSTL(IMM,IMP) + BT(IMM,2)*ECR(IMP,2)
      BSTL(IMM,IMP) = BSTL(IMM,IMP) + BT(IMM,3)*ECR(IMP,3)
      CONTINUE
  370 CONTINUE
  CONTINUE
  DO 360 IMP = 1, 3
    IF (DABS(BSTL(IMM,IMP)) .LE. 0.01) BSTL(IMM,IMP) = 0.0
  360 CONTINUE
  DO MX = 1, 3
    ALI = BSTL(IMM,1)
    AMI = BSTL(IMM,2)
    ANI = BSTL(IMM,3)
  370 CONTINUE
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
AH1 = BSTL(1,3)
AL2 = BSTL(2,1)
AM2 = BSTL(2,2)
AH2 = BSTL(2,3)
AL3 = BSTL(3,1)
AM3 = BSTL(3,2)
AH3 = BSTL(3,3)
PEPMAX = HISTO(L,41,J)
PEPMID = HISTO(L,42,J)
PEPMIN = HISTO(L,43,J)
PSTL1 = PEPMAX*AL1*AH1 + PEPMID*AM1*AH1 + PEPMIN*AM1*AH1
1
PSTL2 = PEPMAX*AL2*AL2 + PEPMID*AM2*AM2 + PEPMIN*AM2*AH2
1
PSTL3 = PEPMAX*AL3*AL3 + PEPMID*AM3*AM3 + PEPMIN*AM3*AH3
1
IF (TESTF(L,ISP2,J) .EQ. 0) THEN
    IF (PSTL2 .LE. TESTF(L,9+ISPG,J) .AND. EPSTL2
        .GT. TESTF(L,9+ISPG,J)) THEN
        CHKGS = .TRUE.
        CSQVER = 999.0
        WRITE (*, *) 'TSTIF 5 999 ELHUM', ELHUM
    ENDIF
    CQHVER = 999.0
    WRITE (*, *) 'TSTIF 6 999 ELHUM', ELHUM
    ENDIF

CHST12 = 0.0
CHST23 = 0.0
CHST13 = 0.0
IF (TESTF(L,ISP,J).EQ.1.0 .AND. TESTF(L,ISP2,J)
    .EQ.1.0) THEN
    CHST12 = DSQRT(TESTF(L,24+ISPG,J)/HISTO(L,26,
        J)*TESTF(L,24+ISPG2,J)/HISTO(L,26,J))
    ELSE IF (TESTF(L,ISP,J).EQ.1.0 .AND. TESTF(L,
        ISP2,J).EQ.0.0) THEN
        CHST12 = DSQRT(TESTF(L,24+ISPG,J)/HISTO(L,26,
            J)*HISTO(L,20,J)/HISTO(L,26,J))
    ENDIF

    IF (EPSTL2.GT.TESTF(L,9+ISPG,J)) CHST12=0.0
    ELSE IF (TESTF(L,ISP,J).EQ.0.0 .AND. TESTF(L,
        ISP2,J).EQ.0.0) THEN
        CHST12 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J))
    ENDIF

    IF (EPSTL2.GT.TESTF(L,9+ISPG,J) .OR. EPSTL1
        .GT. TESTF(L,9+ISPG,J)) CHST12 = 0.0
    ENDIF

IF (TESTF(L,ISP,J).EQ.1.0 .AND. TESTF(L,ISP3,J)
    .EQ.1.0) THEN
    CHST13 = DSQRT(TESTF(L,24+ISPG,J)/HISTO(L,26,
        J)*TESTF(L,24+ISPG3,J)/HISTO(L,26,J))
    ELSE IF (TESTF(L,ISP,J).EQ.0.0 .AND. TESTF(L,
        ISP3,J).EQ.0.0) THEN
        CHST13 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J))
    ENDIF

    IF (EPSTL2.GT.TESTF(L,9+ISPG,J) .OR. EPSTL1
        .GT. TESTF(L,9+ISPG,J)) CHST13 = 0.0
    ENDIF

    IF (TESTF(L,ISP,J).EQ.1.0 .AND. TESTF(L,ISP2,J)
        .EQ.1.0) THEN
        CHST12 = DSQRT(TESTF(L,24+ISPG,J)/HISTO(L,26,
            J)*TESTF(L,24+ISPG2,J)/HISTO(L,26,J))
    ELSE IF (TESTF(L,ISP,J).EQ.0.0 .AND. TESTF(L,
        ISP2,J).EQ.0.0) THEN
        CHST12 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J))
    ENDIF

    IF (EPSTL2.GT.TESTF(L,9+ISPG,J) .OR. EPSTL1
        .GT. TESTF(L,9+ISPG,J)) CHST12 = 0.0
    ENDIF

    IF (TESTF(L,ISP,J).EQ.1.0 .AND. TESTF(L,ISP3,J)
        .EQ.1.0) THEN
        CHST13 = DSQRT(TESTF(L,24+ISPG,J)/HISTO(L,26,
            J)*TESTF(L,24+ISPG3,J)/HISTO(L,26,J))
    ELSE IF (TESTF(L,ISP,J).EQ.0.0 .AND. TESTF(L,
        ISP3,J).EQ.0.0) THEN
        CHST13 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J))
    ENDIF

    IF (EPSTL2.GT.TESTF(L,9+ISPG,J) .OR. EPSTL1
        .GT. TESTF(L,9+ISPG,J)) CHST13 = 0.0
    ENDIF

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHST13 = DSQRT(TDISTF(L,24+ISP3,J)/HISTO(L,26,J)*HISTO(L,20,J)/HISTO(L,26,J))
IF (EPSTL3 GT TEHSTF(L,9+ISP3,J)) CHST13 = 0.0
ELSE IF (TEHSTF(L,ISP,J) EQ 0.0 .AND. TEHSTF(L,ISP3,J) EQ 1.0) THEN
CHST13 = DSQRT(TDISTF(L,24+ISP3,J)/HISTO(L,26,J)*HISTO(L,20,J)/HISTO(L,26,J))
ELSE IF (TEHSTF(L,ISP,J) EQ 0.0 .AND. TEHSTF(L,ISP3,J) EQ 0.0) THEN
CHST13 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J))
ELSE IF (TEHSTF(L,ISP,J) GT TEHSTF(L,9+ISP3,J) OR EPSTL1 GT TEHSTF(L,9+ISP3,J)) CHST13 = 0.0
ELSE IF (TEHSTF(L,ISP3,J) EQ 1.0 .AND. TEHSTF(L,ISP2,J) EQ 1.0) THEN
CHST23 = DSQRT(TDISTF(L,24+ISP2,J)/HISTO(L,26,J)*HISTO(L,20,J)/HISTO(L,26,J))
ELSE IF (TEHSTF(L,ISP3,J) EQ 1.0 .AND. TEHSTF(L,ISP2,J) EQ 0.0) THEN
CHST23 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J))
ELSE IF (TEHSTF(L,ISP3,J) GT TEHSTF(L,9+ISP3,J)) CHST23 = 0.0
ENDIF
ENDIF
ENDIF
TEHP11(2,1) = CHST12*HISTO(L,26,J)*HISTO(L,27,J)
TEHP11(1,2) = CHST12*HISTO(L,26,J)*HISTO(L,27,J)
TEHP11(3,1) = CHST13*HISTO(L,26,J)*HISTO(L,27,J)
TEHP11(1,3) = CHST13*HISTO(L,26,J)*HISTO(L,27,J)
TEHP11(2,3) = CHST23*HISTO(L,26,J)*HISTO(L,27,J)
CALL DTRANS(TEHP11,WT,IOUT)
ICOUPI = 1
CALL DTRANS(TEHXYZ,WT,IOUT)
DO 410 JJ = 1, 6
DO 400 JE = 1, 6
DSTIF(1,J) = TEHXYZ(1,J,JK) + DSTIF(1,J,JK)
400 CONTINUE
410 CONTINUE
ENDIF
420 CONTINUE
RETURN
END

C----------------------------------------  A B O V E  ----------------------------------------
C INCLUDE(PROCESS)
SUBROUTINE ARANGE(ICRACK,NCRACK,IG,IL)
IMPLICIT REAL*8 (A-E,O-Z)
C...SWITCHES: RESUMB=100:10,FORMAT=900:10
C...SWITCHES:
SUBROUTINE AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C COMMON/BLOCKS/HISTa(27,70,15),CRACK(27,250,15),IHISTY,IHIST1
$ ,IHIST2
C
SUBROUTINE TO ELIMINATE INFORMATION REGARDING A VERY
STIFF CRACK WHICH IS COMPLETELY CLOSING. THE ARRAY
'CRACK' WILL BE REARRANGED BY POPPING UP THE DATA
FOR THE REMAINING CRACKS.
C
CRACK(IG,1,IL) = CRACK(IG,1,IL) - 1.0
DO 100 I = ICRAK, NCRAK - 1
C
CRACK INITIATION VALUE
CRACK(IG,37+I,IL) = CRACK(IG,38+I,IL)
C
CRACK DIRECTION
CRACK(IG,137+I,IL) = CRACK(IG,146+I,IL)
CRACK(IG,138+I,IL) = CRACK(IG,147+I,IL)
CRACK(IG,139+I,IL) = CRACK(IG,148+I,IL)
CRACK(IG,140+I,IL) = CRACK(IG,149+I,IL)
CRACK(IG,141+I,IL) = CRACK(IG,150+I,IL)
CRACK(IG,142+I,IL) = CRACK(IG,151+I,IL)
CRACK(IG,143+I,IL) = CRACK(IG,152+I,IL)
CRACK(IG,144+I,IL) = CRACK(IG,153+I,IL)
CRACK(IG,145+I,IL) = CRACK(IG,154+I,IL)
C
OLD STRAIN OF THE CRACK
CRACK(IG,67+I,IL) = CRACK(IG,68+I,IL)
C
CURRENT STRAIN OF THE CRACK
CRACK(IG,67+I,IL) = CRACK(IG,68+I,IL)
C
DIRECT STIFFNESS OF THE CRACK
CRACK(IG,67+I,IL) = CRACK(IG,88+I,IL)
C
SHEAR STIFFNESS OF THE CRACK
CRACK(IG,97+I,IL) = CRACK(IG,98+I,IL)
CRACK(IG,107+I,IL) = CRACK(IG,108+I,IL)
C
STATE OF THE CRACK
CRACK(IG,117+I,IL) = CRACK(IG,118+I,IL)
C
SHEAR STRAIN OF THE CRACK
CRACK(IG,67+I,IL) = CRACK(IG,68+I,IL)
CRACK(IG,77+I,IL) = CRACK(IG,78+I,IL)
C
TERMINATION STRAIN FOR THE CRACK
CRACK(10,127+1,IL) = CRACK(10,128+1,IL)
100 CONTINUE
RETURN
END

C
C
C ------------------------------------------ CRACK ------------------------------------------
C
C INCLUDE(PROCESS)
SUBROUTINE CRACKEP(SRACK, PROPER, ES, ICRACK, IGAUSS, ILAYER)
IMPLICIT REAL*8(A-H,0-Z)
C...SWITCHES: RESUMB=100:10,FORMAT=900:10
C...SWITCHES: 
C-----------------------------------------------------------------------------------------------
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBRoutines OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C----------------------------------------------------------------------
COMMOM/BLOCKS/HIST0(C27,70,15),CRACK(C27,250,15),IHISTORY,IHISTORY1
$IHISTORY2
DIMENSION PROPER(25)
C
C
SUBROUTINE TO DETERMINE SECANT YOUNG'S MODULUS CORRESPONDING TO THE RECENT STRAIN ACROSS THE CRACK, 'EPSCRK'.
C
C
C RETRIEVE THE CRACK INITIATION STRESS, FT, AND THE CRACK TERMINATION STRAIN, EPSNOT .
C
C
FT = CRACK(IGAUSS,37+ICRACK,ILAYER)
EPSNOT = CRACK(IGAUSS,127+ICRACK,ILAYER)
C
FOR CRACK TREATMENT RELATED TO TENSION-STIFFENING ...

IF (PROPER(16) .GT. 0.0) GO TO 100
C
CRACKING RELATED TO MATERIAL SOFTENING MODEL
C
TRANSIT = STRAIN AT THE TRANSITION POINT FOR BILINEAR STRAIN SOFTENING MODEL.
C
TRANSIT = EPSNOT*2.9.
C
..... DETERMINE STEP FUNCTIONS FOR BILINEAR STRAIN SOFTENING.
C
S1 = 0.0
S2 = 0.0
IF (EPSCRK.GT.0. AND. EPSCRK.LE.TRANSIT) S1 = 1.0
IF (EPSCRK.GT.TRANSIT .AND. EPSCRK.LT.EPSNOT) S2 = 1.
ES = 3.*FT*(S1*(-7.+EPSNOT/(3.*EPSCRK)))+S2*(-1.+EPSNOT/EPSCRK
1 .))/EPSNOT
RETURN
C
C
** TENSION STIFFENING CRACK **
100 CONTINUE
S = 0.0
IF (EPSCKR.GT.0.0 .AND. EPSCKR.LE.EPSHOT) S = 1.0
ES = S*(FT*(1./EPSCKR-1./EPSHOT))
RETURN
END

-------------------------------------------------------------------------
INCLUDE(PROCESS)
SUBROUTINE DETERMINE SECANT YOUNG'S MODULUS CORRESPONDING TO THE RECENT STRESS ACROSS THE CRACK, 'SIGCRK'.
RETRIEVE THE CRACK INITIATION STRESS, FT, AND THE CRACK TERMINATION STRAIN, EPSHOT .
FT = CRACK(IGAUSS,37+ICRACK,ILAYER)
EPSHOT = CRACK(IGAUSS,127+ICRACK,ILAYER)
FOR CRACK TREATMENT RELATED TO TENSION-STIFFENING ...
IF (PROPER(16).GT.0.0) GO TO 100
CRACKING RELATED TO MATERIAL SOFTENING MODEL
..... DETERMINE STEP FUNCTIONS FOR BILINEAR STRAIN SOFTENING.
S1 = 0.0
S2 = 0.0
IF (SIGCRK.LT.FT .AND. SIGCRK.GE.FT/3.0) S1 = 1.0
IF (SIGCRK.GE.0.0 .AND. SIGCRK.LT.FT/3.0) S2 = 1.0
ES = (3.*FT+SIGCRK/EPSHOT)*(S1/(FT-SIGCRK)+S2/(3.*FT-7.*SIGCRK))
RETURN
** TENSION STIFFENING CRACK **
100 CONTINUE
S = 0.0
IF (SIGCRK.GE.0.0 .AND. SIGCRK.LT.FT) S = 1.0
ES = S*FT*SIGCRK/(EPSHOT*(FT-SIGCRK))
RETURN
END

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SUBROUTINE CRKCUN(PROPER, DCRACK, ILAYER, ICRACK, IGAUSS, EPSCRK, ES, CHKGUS, HSTIFF, CONVER, ILUTER, EPSOLD, IFLAG)

IMPLICIT REAL*8 (A-H, O-Z)

C..SWITCHES: RENUMB=100:10, FORMAT=800:10
C..SWITCHES:

SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

C
C CRKSTN
C
C******#****#**************#*,** *******  #**#*»****#******

COMMON/BLOCK6/HISTO(27,70,15), CRACK(27,250,15), IHISTY, IHIST1, IHIST2
DIMENSION PROPER(26)
LOGICAL CHKGUS

C THE CRACK HAD BEEN OVER-SOFTENED IN THE PREVIOUS ITER...

C IF (DCRACK .LT. ES) DIFFER = (ES-DCRACK)*100./ES
AREA = 0.0
C
C ........ DETERMINE THE STRAIN CORRESPONDING TO CRACK STIFFNESS
C 'DCRACK' EMPLOYED IN THE CURRENT ITERATION.

CALL CRKSTN (PROPER, DCRACK, EPSOLD, ILAYER, ICRACK, IGAUSS, ILUTER)
SIGOLD = DCRACK*EPSOLD
SIGNEW = ES*EPSCRK
FT = CRACK(IGAUSS, 37+ICRACK, ILAYER)
EPSNOT = CRACK(IGAUSS, 127+ICRACK, ILAYER)

AREA = AREA UNDER THE STRESS-STRAIN DIAGRAM OF THE
CURRENT CRACK INDICATING THE AMOUNT OF ENERGY
TO BE CONSUMED FOR COMPLETE OPENING OF THE CRK.

............... MATERIAL SOFTENING CRACK

C IF (PROPER(16) .LT. 0.0) AREA = 5.0*FT*EPSNOT/18.0
C
C .... CRACK RELATED TO TENSION STIFFENING

C IF (PROPER(16) .GT. 0.0) AREA = .50*FT*EPSNOT
C
C COMPARE THE DIFFERENCE IN ENERGY ABSORPTION, DETERMINED
BY THE AREA ENCLOSED BY THE EMPLOYED AND THE UPDATED
CRACK STIFFNESS, WITH THE TOTAL ENERGY OR AREA UNDER
THE STRESS-STRAIN CURVE OF THE CURRENT CRACK.

SUBARA = .50*FT*(EPSCRK-EPSOLD)
IF (PROPER(16) .LT. 0.0) THEN
IF (SIGOLD.GE.FT/3.0 .AND. SIGNEW.LE.FT/3.) SUBARA = AREA -
1 .60*(FT*EPSOLD-SIGNEW*EPSNOT)
IF (SIGOLD.LE.FT/3.0 .AND. SIGDEW.LE.FT/3.0) SUBARA = .50*EPSHOT*(SIGOLD-SIGDEW)
ENDIF

DIFFER = 100.0*(SUBARA/AREA)
IF (DIFFER .LE. TOLER) RETURN
IF (DIFFER .GT. TOLER) THEN

UPDATE THE CRACK STIFFNESS

CRACK(IGAUSS,87+ICRACK,ILAYER) = ES
IFLAG = 1
CHGUSS = .TRUE.
NSTIFF = 1
CONVER = 999.00
WRITE (*(, *)) 'CRACK 999 ELNUM', 'CRACK', ICRACK
ENDIF
RETURN
END

----------------------------------------------------- CRKS TB -----------------------------------------------------

SUBROUTINE CRKSTH(PROPER,DCRACK,EPDCRK,ILAYER,ICRACK,IGAUSS,TOUT)
IMPLICIT REAL*8 (A-H.D-Z)

C...SWITCHES: REHUMB=100:10,FORKAT=900:10
C..-SWITCHES:

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT

COMMD/BL0CK5/HIST0(27,70.15),CRACKC27,250,15).IHISTY,IHIST1 $,IHIST2
DIMENSION PROPER(25)

SUBROUTINE TO DETERMINE THE STRAIN ACROSS A CRACK FOR A GIVEN CRACK STIFFNESS.

FT = CRACK(IGAUSS,37+ICRACK,ILAYER)
EPSHOT = CRACK(IGAUSS,127+ICRACK,ILAYER)

FOR TENSION STIFFENING CRACK

IF (PROPER(16) .GT. 0.0) GO TO 100

......... CRACKING RELATED TO MATERIAL STRAIN SOFTENING

DETERMINE STEP FUNCTIONS FOR BILINEAR STRAIN SOFTENING

S1 = 0.0
S2 = 0.0

ETRANS = CRACK STIFFNESS AT THE BILINEAR TRANSITION PNT

ETRANS = 1.50*FT/EPShOt
IF (DCRACK .GE. ETRANS) SI = 1.0
IF (DCRACK .LT. ETRANS) S2 = 1.0
EPDCRK = FT*EPSN0T*(SI/(3.0*FT+DCRACK*EPSN0T)+3.*S2/(3.*FT+7.*DCRACK*EPSSDT))
RETURN

C FOR TENSION STIFFENING CRACK
C
100 CONTINUE
EPDCRK = FT*EPSN0T/(DCRACK*EPSN0T+FT)
RETURN
END

C INCLUDE(PROCESS)
SUBROUTINE OVERFT(EPSOLD, TOLER, IGAUSS, ICRACK, ILAYER, PROPER, DCRACK, CHKGUS, HSTIFF, CONVER, IOUT, SIGCRK, ES, IELEM)
IMPLICIT REAL*8 (A-H,0-Z)
C...SWITCHES: RENUHB=100:10,FORHAT=900:10
C...SWITCHES:
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C CRACKE
C
COMMON/BLOCK5/HISTO(27,70,16),CRACK(27,250,15),IHISTY,IHIST1 $,IHIST2
DIMENSION PROPER(2B)
LOGICAL CHKGUS
C
SUBROUTINE TO CORRECT OVERSHOOTING BEYOND THE CRACKING STRESS FOR THE CURRENT CRACK. EVEN THOUGH THE CALCULATE
'S' BASED ON THE CURRENT STRAIN ACROSS THE CRACK DID NO
SHOW THE NEED FOR UPDATING THE CRACK STIFFNESS THE EWER
GY CRITERION FOR SOFTENING THE CRACK AND HENCE ELIMINAT
ING THE OVERSHOOTING MUST BE USED.
C
.... GET THE INITIATION STRESS AND THE TERMINATION STRAIN.

FT = CRACK(IGAUSS,37+ICRACK,ILAYER)
EPSSHOT = CRACK(IGAUSS,127+ICRACK,ILAYER)
IF (SIGCRK.GT.FT .AND. ES.GT.DCRACK) THEN
    WRITE(*,900)ICRACK,IGAUSS,ILAYER,SIGCRK,FT,DCRACK,ES,IELEM
    STOP
ENDIF

Determine a new strain on the stress-strain curve
of the crack. The area enclosed by the old and the
new crack strain is equal to the tolerance.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
339

FOR TENSION STIFFENING CRACK

IF (PROPER(16) .GT. 0.0) EPSNEW = EPSOLD + .010*TOLER*EPSHOT
IF (PROPER(16).LT.0.0 .AND. EPSOLD.LT.2.*EPSHOT/9.0) THEN
EPSNEW = EPSOLD + .010*TOLER*EPSHOT/9.0
ENDIF

WITH THE NEW CRACK STRAIN CALCULATE ITS CORRESPONDING SECANT 'E'.

CALL CRACK (EPSNEW, PROPER, ESNEW, ICRACK, IGAUSS, ILAYER, IOUT)
EINITL = PROPER(1)
ESNEW = DMAXKEINITL/1000.0,ESNEW)

UPDATE THE CRACK STIFFNESS

CRACK(IGAUSS,87+ICRACK, ILAYER) = ESNEW
CEKGS = .TRUE.
BSSTIFF = 1
COVER = 999.
RETURN

900 FORMAT(IX,' ERROR IN SUBROUTINE CRKPHT',/,'ST',
1 'FOR CRACK NO. ',I3,' LOCATED IN GAUSS POINT ',I3,
2 ' OF LAYER ',I3,' IN THE CURRENT ELEMENT ',I3,
3 '/,'STRESS VALUE=','D12.5,'/,'EMPLOYED CRACK',
4 'STIFFNESS=','D12.6,'< UPDATED CRACK ',
5 'STIFFNESS=','D12.6,'>,ELHUM',I2)

END

...SWITCHES: REHUHB=100:10,FORMAT=900:10
...SWITCHES:

SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

INTEGER ELHUM
REAL*8 LTHICK
COMMON/BLQCKB/HISTO(27,20,1B),CRACK(27,20,1B),IHISTY,IHIST1
COMMON/TEHSTF(27,250,18),IHISTY,IHIST1
COMMON/TEHSTF,CONVER,ING,ICRACK,IGAUSS,EII,IFG)
REAL*8 PROPER(26)
LOGICAL CHEKUS
IF (ECRAC .EQ. 1) THEN
RECOVER THE OLD STRAINS,STIFFNESS,STATE FOR THE CURRENT CRACK
EPS0 = TESSTF(IGAUSS,15+ICRACK,ILAYER)
STATE = TESSTF(IGAUSS,21+ICRACK,ILAYER)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SPRING = TESTSF(IGAUSS,24+ICRACK,ILAYER)
SIGHT = SPRING*EPSH
EINITL = PROPER(1)
CRESTL = TESTSF(IGAUSS,9+ICRACK,ILAYER)
CLOSED SPRING @EPSH(CRACK OPENING)

IF (EPSH.GT.0.0 .AND. STATE.EQ.4.0) THEN
EST = TESTSF(IGAUSS,27+ICRACK,ILAYER)
ELSE IF STIFFNESS LESS THEN 10000TH THAT OF CONCRETE SET TO THAT!
   IF (EST .LT. EINITL/1000.) EST = EINITL/1000.0
   TESTSF(IGAUSS,24+ICRACK,ILAYER) = EST
   TESTSF(IGAUSS,18+ICRACK,ILAYER) = CRESTL
   TESTSF(IGAUSS,21+ICRACK,ILAYER) = 3.0
   CHKUS = .TRUE.
   ESTIFF = 1
   COVER = 999.0
   WRITE (*, •) 'SPGH 1 999 ELSUM', ELSUM
   GO TO 100
ENDIF

UNLOADING AND RELOADING CRACKS SET TO LOADING CRACKS

IF (EPSH.GT.0.0 .AND. EPSH.GE.EPSO) THEN
IF (STATE.EQ.2.0 .OR. STATE.EQ.3.0) THEN
   TESTSF(IGAUSS,21+ICRACK,ILAYER) = 1.0
   TESTSF(IGAUSS,18+ICRACK,ILAYER) = EPSH
   TESTSF(IGAUSS,12+ICRACK,ILAYER) = EPSH
ELSE IF STIFFNESS LESS THEN 10000TH THAT OF CONCRETE SET TO THAT!
   IF (EPSH .LE. EINITL/1000.0) RETURN
   CALL CKTEH(SPRIHG,EST,IGAUSS,ICRACK,ILAYER,ELSUM,HEL,IFG)
ENDIF

CHECK CONVERGENCE OF THE STIFFNESS

IFLAG = 0
FT = TESTSF(IGAUSS,6+ICRACK,ILAYER)
CALL SPCON (SPRING, ILAYER, ICRACK, IGAUSS, EPSH, EST,
1  CHKUS, ESTIFF, COVER, TOLER, IGU, EPSO, IFLAG,
2  ELSUM, HEL, IFG)
IF (COVER .EQ. 999) WRITE (*, •) 'SPGH 2 999 ELSUM', ELSUM
GO TO 100
ENDIF

WRITE(*,*) 'OVRFT SIGYLD',SIGYLD,'SRATIO',SRATIO,'RHD',RHO,'LY',ILL
**EPSCR1** = SRATIO*FT*EPSY/SIGYLD

**EPSCR2** = (SRATIO*FT/SIGYLD)*(1.0+(1.0+RHO*SRATIO)/(10.0+RHO*SRATIO))*EPSY

**EPSCR3** = (1.0-(1.0/(2.0*RHO*SRATIO))+(SRATIO*FT/SIGYLD))*EPSY

**EPSCR4** = EPSY

WRITE(*,*),' EPS1', EPSCR1, ' EPS2', EPSCR2, ' EPS3', EPSCR3, ' EPS4', EPSCR4

C WRITE C  *, * )  '  EPS1  EPS2  EPS3  EPS4

C = 560.0

FT1 = FT

FT2 = (1.0-(1.0+RHO*SRATIO)/10.0)*FT*DEXP((-C*(EPSCR2-EPSCR1)))

FT3 = FT*DEXP((-C*(EPSCR3-EPSCR1)))/2.0

FT4 = FT/10.0

IF (EPSO .LE. EPSCR2) THEN
  IF (SIGHT .LE. FT1) GO TO ICO
ELSE IF (EPS0.GT.EPSCR2 .AND. EPS0.LE.EPSCR3) THEN
  IF (SIGHT .LE. FT2) GO TO 100
ELSE IF (EPS0.GT.EPSCR3 .AND. EPS0.LE.EPSCR4) THEN
  IF (SIGHT .LE. FT3) GO TO 100
ENDIF
ENDIF
ENDIF
CALL OVRFT (EPS0, TOLER, IGAUSS, ICRACK, ILAYER, EINITL, SPRING, CHKGUS, NSTIFF, CONVER, IOUT, SIGHT, EST, ELHUH, HHEL, IFG)
GO TO 100
ENDIF
ENDIF

C PARTIAL OPENING OR CLOSING OF CRACK

IF (EPSH.GT.0.0 .AND. EPSH.LT.EPSO) THEN
  PEPSN = TEHSTF(IGAUSS,18+ICRACK,ILAYER)
  IF (EPSH .LT. PEPSN) THEN
    TEHSTF(IGAUSS,21+ICRACK,ILAYER) = 2.0
  ELSE
    TEHSTF(IGAUSS,21+ICRACK,ILAYER) = 3.0
  ENDIF
  TEHSTF(IGAUSS,18+ICRACK,ILAYER) = EPSH
  GO TO 100
ENDIF

C COMPLETE CLOSING OF CRACK

IF (EPSH.LT.0.0 .AND. STATE.HE.4.0) THEN
  TEHSTF(IGAUSS,21+ICRACK,ILAYER) = 4.0
  TEHSTF(IGAUSS,27+ICRACK,ILAYER) = TEHSTF(IGAUSS,24+ICRACK,ILAYER)
ENDIF

C 100 CONTINUE
RETURN
END

C******************************************************************************

C INCLUDE(PROCESS)

SUBROUTINE OVRFT(EPS0,TOLER,IGAUSS,I,ILAYER,EINITL,$
SPRING,CHKGUS,STIFF,CONVER,IOUT,$
SIGHT,EST,ELHUH,HHEL,IFG)

IMPLICIT REAL*8 (A-H,O-Z)

C...SWITCHES: AENUM=100:10,FORMAT=900:10
C...SWITCHES:
SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

INTEGER ELNUM
REAL*8 LTHICK
COMMON/BLOCK/HISTO(27,70,15),CRADE(27,250,15),IHISTY,IHIST1
$I,IHIST2
COMMON/ABC/TESTF(27,80,16),ITSPG,ITSG1,ITSG2
DIMENSION LTHICK(9),ZS(9),DCS(3,3)
LOGICAL CHKGUS

SUBROUTINE CORRECTS OVERSHOOTING OF STRESS IN SPRING1 DIRECTION

IF (SIGHT.GT.1.10+FT .AND. DABS(EST).GT. DABS(SPRING)) THEN
WRITE (*,*) 'EST', EST, 'SPRING', SPRING, 'ELHUM', ELHUH
WRITE (*,*) 'ILAYER', ILAYER, 'IGAUSS', IGAUSS
WRITE (*, 900)
STOP
ENDIF

IFG=2
IF (IFG .EQ. 1) THEN
FT1 = 0.35+FT
FT2 = 0.10+FT
EPSCR = CKSTIF
EPSCR1 = EPSY*0.80
EPSCR2 = EPSY
AREA = 0.5*(EPSCR1-EPSCR)*(FT1+FT)+0.5*(EPSCR2-EPSCR1)*(FT2+FT)
EPSTN = EPSO + TOLER+AREA/(FT*0.5*300.0)
ELSE IF (IFG .EQ. 2) THEN
SIGYLD = TESTF(IGAUSS,43,ILayer)
SRATIO = TESTF(IGAUSS,60,ILayer)
ILL = TESTF(IGAUSS,66,ILayer)
CALL LYIHFO (ILL, LTHICK, ZS, MATRL, DCS, HHEL, ELNUM, NIPXI,
1 CALL GETTHK (L, ELNUM, HHEL, THICKE, RAD)
CALL GETTHK (L, ELNUM, HHEL, THICKE, RAD)
RHO = THICK1/THICKE
WRITE(*,*)'OVRFT SIGYLD',SIGYLD, 'SRATIO',SRATIO, 'RHO',RHO, 'LY',ILL
EPSCR1 = SRATIO*FT+EPSY/SIGYLD
EPSCR2 = (SRATIO*FT/SIGYLD+1.0*(1.0+RHO*SRATIO))/(1.0+RHO*
1 SRATIO)) EPSY
EPSCR4 = (1.0-(1.0/(2.*RHO*SRATIO)))*(SRATIO*FT/SIGYLD)) EPSY
WRITE(*,*)'EPSCR1',EPSCR1, 'EPSCR2',EPSCR2, 'EPSCR3',EPSCR3
WRITE(*,*)'EPSCR4',EPSCR4
C = 550.0
FT1 = FT
FT2=(1.0-(1.0+RHO*SRATIO)/10.0)*FT+EXP((-C*(EPSCR2-EPSCR1))
FT3 = FT*EXP((-C*(EPSCR3-EPSCR1))/2.0)
FT4 = FT/10.0
WRITE(*,*)'FT1', 'FT2', 'FT3', 'FT4'
AREA = 0.5*(FT1-FT2)*(EPSCR2-EPSCR1) + (EPSCR2-EPSCR1)*FT2 + 
1.0*(FT2-FT3)*(EPSCR3-EPSCR2) + (EPSCR3-EPSCR2)*FT3 + 0.5* 
(FT3-FT4)*(EPSCR4-EPSCR3) + (EPSCR4-EPSCR3)*FT4
IF (EPSO .LT. EPSCR2) THEN
EPSTD = EPSO + TOLER*AREA/(FT1*100.0)
EPSTH = DMISI(EPSCR2,EPSTH)
ELSE IF (EPSO .GE. EPSCR2 .AND. EPSO .LE. EPSCR3) THEN
EPSTH = EPSO + TOLER*AREA/(FT2*0.5*100.0)
EPSTH = DMISI(EPSCR3,EPSTH)
ELSE IF (EPSO .GE. EPSCR3 .AND. EPSO .LE. EPSCR4) THEN
EPSTH = EPSO + TOLER*AREA/(FT3*0.5*100.0)
EPSTH = DMISI(EPSCR4,EPSTH)
ENDIF
C FIND NEW STIFFNESS
CALL CTEN (EPSTH, ESTH, IGAUSS, I, ILAYER, ELNUM, NHEEL, IFG)
C ----- IF STIFFNESS LESS THAN 1000TH THAT OF CONCRETE SET TO THAT
IF (DABS(ESTH) .LE. ESTH/1000.0) ESTH = ESTH/1000.0
TEHSTF(IGAUSS,24+1,ILAYER) = ESTH
TEHSTF(IGAUSS,15+1,ILAYER) = ESTH
TEHSTF(IGAUSS,18+1,ILAYER) = ESTH
COVYT = 999.0
NSTIFF = 1
CHEGUS = .TRUE.
RETURN
900 FORMAT(IX, ' ERROR IN SPG.OVERSHOOT: HIGHER STIFFNESS ENCOUNTERED')
END
C******************************************************************************
C INCLUDE(PROCESS)
SUBROUTINE SPCON(SPRING, ILAYER, I, IGAUSS, EPSU, EST,
$ CHKGUS, ESTH, COVYT, TOLER, IOUT, EPSO,
$ IFLAG, ELNUM, NHEEL, IFG)
IMPLICIT REAL*8(A-H,0-Z)
C* SWICHES: RESTUMB=100:10,FORMAT=900:10
C* SWICHES:
C******************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C SPSTH LYPINO GTIJK GETTHR
C******************************************************************************
REAL*8 LTHICK
INTEGER ELNUM
COMMON/BLOK/3HISTO(27,70,1S),CRACK(27,260,1S),IHISTY, IHISS1
$ , IHISS2
COMMON/ABC/TEHSTF(27,60,1S),IT2SPO,ITBSG1,ITBSG2
DIMENSION LTHICK(9),ZS(9),DCS(3,3)
LOGICAL CHEGUS
IF (SPRING .LT. EST) THEN
C CRACK SPRING HAS BEEN OVERSOFTENED IN PREVIOUS ITERATION
C DIFFER = (EST-SPRING)*100/EST
WRITE (*,900) I, IGAUSS, ILAYER, SPRING, EST, DIFFER
C END IF

EHDIF

C FIND STRAIN CORRESPONDING TO SPRING

C FT = THESTF(IGAUSS,1*6,ILAYER)
C CKSTIM = THESTF(IGAUSS,1*6,ILAYER)
C EPSY = THESTF(IGAUSS,12*1,ILAYER)
C
C IF (SPRING .GT. EST) THEN
C
CALL SPSTF(SPRING,EP50,ILAYER,1,IGAUSS,EP50,EL5UM,H5EL,IFG)
C
C AREA = 0
C IF (IFG .EQ. 1) THEN
FT1 = 0.3*FT
FT2 = 0.1*FT
EPSCR = CKSTIM
EPSCR1 = EPSY*0.50
EPSCR2 = EPSY
IF (EP50 .LE. EPSCR1) THEN
SUBARA = 0.5*FT+(EP50-EP50)
ELSE IF (EP50 .LE. EPSCR1 .AND. EP50 .LE. EPSCR1) THEN
SUBARA = 0.5*EP50*EP50
ENDIF

AREA = 0.5*FT1 + 0.5*FT2
C ELSE IF (IFG .EQ. 2) THEN
SIGYLD = THESTF(IGAUSS,1*63,ILAYER)
SRATIO = THESTF(IGAUSS,1*60,ILAYER)
ILL = THESTF(IGAUSS,1*66,ILAYER)
CALL LYIHFO (ILL, LTHICK, ZS, MATRL, DCS, HHEL, ELHUH,
1 CALL GTLTK (L, ELHUH, HHEL, THICKL, ZSI, LTHICK, ZS)
RHO = THICKL/THICK
EPSCR1 = SRATIO+FT+EPSY/SIGYLD
EPSCR2 = (SRATIO+FT/SIGYLD)*(1.0+(1.0+RHO+SRATIO)/(10.0*
1 RHO+SRATIO))*EPSY
EPSCR3 = (1.0-(1.0/(2.+RHO+SRATIO))+(SRATIO+FT/SIGYLD))+
1 EPSY
EPSCR4 = EPSY
C = 5.50
FT1 = FT
FT2 = (1.0-(1.0+RHO+SRATIO)/(10.0))*FT+DEIP((-C+(EPSCR2-
1 EPSCR1)))
FT3 = FT+DEIP((-C+(EPSCR3-EPSCR1))/2.0)
FT4 = FT/10.0
AREA = 0.5*(FT1+FT2)*(EPSCR2-EPSCR1) + (EPSCR2-EPSCR1)*FT2
+ 0.5*(FT2-FT3)*(EPSCR3-EPSCR2) + (EPSCR3-EPSCR2)*FT3
+ 0.5*(FT3-FT4)*(EPSCR4-EPSCR3) + (EPSCR4-EPSCR3)*FT4
IF (EP50 .LE. EPSCR2 .AND. EP50 .LE. EPSCR2) THEN
SUBARA = 0.5*(EP50-EP50)*FT1
1 .LE. EPSCR3) THEN
SIGE = EST*EP50
SUBARA = 0.5*(EPSCR2-EP50)*FT1 + 0.5*(EPSCR2-EPSCR1)*
1 FT2 + 0.5*(EP50-EPSCR2)*(FT2+SIGE) - 0.5*(EP50-
1 EPSCR2)
ENDIF
ELSE IF (EPSH.GT.EPSCR3 .AND. EPSH.LE.EPSCR4 .AND. EPSO .GT. EPSCR3 AND. EPSO.LE.EPSCR4) THEN
  SIGN = EST*EPSH
  SIGO = SPRING*EPSO
  SUBARA = 0.5*(EPSCR2-EPSRCP1)*(FT2+FT3) + 0.5*(EPSH-EPSG)-0.5*(EPSH-EPSCR3)*SIGO - 0.5*(EPS0-EPSCR3)*(SIGO+FT2) + 0.5*(EPS0-EPSCR3)*SIGO
ELSE IF (EPSH.GT.EPSCR3 .AND. EPSH.LE.EPSCR4 .AND. EPSO .GT. EPSCR3 .ADD. EPSO.LE.EPSCR4) THEN
  SIGN = EST*EPSH
  SIGO = SPRING*EPSO
  SUBARA = 0.5*(EPSCR2-EPSRCP1)*(FT2+FT3) + 0.5*(EPSH-EPSG)-0.5*(EPSH-EPSCR3)*SIGO - 0.5*(EPS0-EPSCR3)*(SIGO+FT2) + 0.5*(EPS0-EPSCR3)*SIGO
ELSE IF (EPSH.GT.EPSCR3 .AND. EPSH.LE.EPSCR4 .AND. EPSO .GT. EPSCR3 .ADD. EPSO.LE.EPSCR4) THEN
  SIGN = EST*EPSH
  SIGO = SPRING*EPSO
  SUBARA = 0.5*(EPSCR2-EPSRCP1)*(FT2+FT3) + 0.5*(EPSH-EPSG)-0.5*(EPSH-EPSCR3)*SIGO - 0.5*(EPS0-EPSCR3)*(SIGO+FT2) + 0.5*(EPS0-EPSCR3)*SIGO
ENDIF

ENDIF

DIFFER = (SUBARA/AREA)*100.0

IF (DIFFER .LE. TOLER) RETURN
IF (DIFFER .GT. TOLER) THEN
  TESTF(IGAUSS,24+1,ILAYER) = EST
  IFLAG = 1
  COVER = 999.0
  NSTIFF = 1
  CHKGUS = .TRUE.
ENDIF

RETURN

FORMAT(1X,'SUB.SPRIHGCOHl.OVERSOFTEHIHG CRACK#',13,
  'FOR GAUSSPOINT',12,'LAYER#',12,'OLDSTF',D12.B,

C
C*****************************************************************************
C INCLUDE(PROCESS)
SUBROUTINE EPSR(SPRIHG,EPSON,ILAYER,1,IGAUSS,EPSPH,ELBUN,IGK,IIFG)
IMPLICIT REAL*8 (A-H,0-Z)
C...SWITCHES: RETNUM=100:10,FORMAT=900:10
C...SWITCHES:
C*****************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C LYIIFG O GTLTK GETTHK
C
C***************************************************************************
INTEGER ELHUM
REAL*8 LTHICK
COMMON/BLOCKS/HISTO(27,70,16),CRACK(27,250,16),IHISTY,IHIST1
$ .IHIST2
COMMON/ABC/TEHSTF(27,80,16),ITHSPG,ITHSG1,ITHSG2
DIMENSION LTHICK(9),ZS(9),DCS(3,3)
C
C FIND THE STRAIN FOR A GIVEN TENSION STIFFNESS SPRING-1
C
FT = TEHSTF(IGAUSS,1=6,IENER)
EPSY = TEHSTF(IGAUSS,12+1,IENER)
CSTH = TEHSTF(IGAUSS,.I+9,IENER)
C IFG=2
IF (IFG .EQ. 1) THEN
FT1 = 0.35+FT
FT2 = 0.10+FT
EPSCR = CSTH
EPSCR1 = EPSY+0.50
EPSCR2 = EPSY
STIF1 = FT1/EPSCR1
C IF (DABS(SPRIHG) .GE. DABS(STIF1)) THEN
EPSO = (FT1*EPSCR1-FT1*EPSCR)/(SPRIHG*(EPSCR1-EPSCR)+(FT1-
1  FT1))
EPSO = DMAX1(CSTH,EPSO)
ELSE
EPSO = (FT1*EPSCR2-FT2*EPSCR1)/(SPRIHG*(EPSCR2-EPSCR1)+(
1  FT1-FT2))
EPSO = DMAX1(0.5+EPSY,EPSO)
ENDIF
ELSE IF (IFG .EQ. 2) THEN
SIGYLD = TEHSTF(IGAUSS,1=63,IENER)
SRATIO = TEHSTF(IGAUSS,1=60,IENER)
ILL = TEHSTF(IGAUSS,1=66,IENER)
CALL LYIFGO (ILL, LTHICK, ZS, MATEL, DCS, NREL, ELHUM, WIPXI,
1  WIPETA, WIPS)
CALL GTLTK (L, ELHUM, BREL, THICK, ZS, LTHICK, ZS)
CALL GETTHK (L, ELHUM, BREL, THICK, RAD)
RHO = THICKL/THICKE
EPSCR1 = SRATIO*FT*EPSY/SIGYLD
EPSCR2 = (SRATIO*FT/SIGYLD)*(1.0+(1.0+RHO+SRATIO)/(10.0*RHO-
1  SRATIO))*EPSY
EPSCR3 = (1.0-(1.0/(2.*RHO+SRATIO)))*EPSY
EPSCR4 = EPSY
C = 550.0
FT1 = FT
FT2=(1.0-(1.0+RHO+SRATIO)/10.0)*FT+DEXP((-C*(EPSCR2-EPSCR1)))
FT3 = FT+DEXP((-C*(EPSCR3-EPSCR1)))/2.0
FT4 = FT/10.0
SPG1 = FT1/EPSCR1
SPG2 = FT2/EPSCR2
SPG3 = FT3/EPSCR3
SPG4 = FT4/EPSCR4
IF (SPRING .GE. SPG2) THEN
EPSO = (FT2*EPSCR1-FT1*EPSCR2)/((FT2-FT1)*(EPSCR1-EPSCR2))
1 SPRING
ELSE IF (SPRING.LT.SP2 .AND. SPRING.GE.SP23) THEN
EPSO = (FT2*EPSCR2-FT3*EPSCR3)/((FT3-FT2)*(EPSCR2-EPSCR3))
1 SPRING
ENDIF
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ELSE IF (SPRIG .LT. SPG3 .AND. SPRIG .GE. SPG4) THEN
    EPSO = (FT4+EPSCR3-FT3+EPSCR4)/((FT4-FT3)+(EPSCR3-EPSCR4))
ENDIF
ENDIF
WRITE(*,*; 'SPGH STIFF*,SPRIG,'EPSOLD=',EPSO
RETURN

C*************************************************************************
C INCLUDE(PROCESS)
SUBROUTINE CKSTIH(EPSH,SPRIG,IGAUSS, I, ILAYER,ELHUM,HHEL,IFG)
C*************************************************************************
C THIS SUBROUTINE CALCULATES THE STIFFNESS OF THE TENSION
C STIFFNESS SPRING AT A GIVEN SAMPLEPOINT OF A GIVEN LAYER
C OF A PARTICULAR ELEMENT.
C
C INPUT PARAMETERS: FT=TENSILE STRESS IN CONCRETE AT SPRING
C INITIATION.
C EPSY=YIELD STRAIN OF ADJACENT STEEL LAYER
C EPST=CURRENT SPRING STRAIN.
C
C OUTPUT RETURNED: SPRING: SECANT STIFFNESS CORRESP. TO EPST.
C
C IMPLICIT REAL=8 (A-H,0-2)
C...SWITCHES: RENUM=100:10,FORMAT=900:10
C...SWITCHES:
C*************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C INTEGER ELNUM
REAL-8 LTHICK
COMMON/ABC/TEHSTF(27,80,16),ITNPSG,ITSPG2,ITSPG3
COMMON/BLOCKS/HISTO(27,70,16),CRACK(27,260,16),IHISTY,IHIST1
$,IHIST2
DIMENSION LTHICK(9),ZS(9),DCSC3,3)
C*************************************************************************
C FIND THE TENSION STIFFNESS OF SPRING FOR GIVEN STRAIN-1
FT = TEHSTF(IGAUSS,I+6,I,ILAYER)
EPSY = TEHSTF(IGAUSS,12+I,I,ILAYER)
CKSTIH = TEHSTF(IGAUSS,X+9,I,ILAYER)
IF (IFG .EQ. 1) THEN
    FT1 = 0.35*FT
    FT2 = 0.10*FT
    EPSCR = CKSTIH
    EPSCR1 = 0.50*EPSY
    EPSCR2 = EPSY
    IF (EPSH .LE. EPSCR1) THEN
        SPRIG = (FT1*(EPSCR1-EPSH)+EPSH*(EPSH-EPSCR1))/EPSH*(EPSH-EPSCR1)
    ELSE
        SPRIG = (FT1*(EPSCR2-EPSH)+FT2*(EPSH-EPSCR1))/EPSH*(EPSH-EPSCR1)
    ENDIF
    ELSE IF (IFG .EQ. 2) THEN
        SIGYLD = TEHSTF(IGAUSS,1+63,I,ILAYER)
        ...
SRATIO = TEBSTF(IGAUSS,1+60,ILAYER)
ILL = TEBSTF(IGAUSS,1+66,ILAYER)
CALL LLYIF (ILL, LTHICK, ZS, MATRL, DCS, NNEL, ELSUM, HIPXI,
1
EPSETA, EPSI)
CALL GILTK (L, ELNUM, NNEL, THICKL, ZSI, LTHICK, ZS)
CALL GETSHK (L, ELNUM, NNEL, THICK, RAD)
RHO = THICKL/THICKE
EPSCR1 = SRATIO+FT+EPSY/SIGYLD
EPSCR2 = (SRATIO*FT/SIGYLD)*(1.0+(1.0+RHO+SRATIO)/(10.0+RHO+
1 SRATIO))*EPSY
EPSCR3 = (1.0-(1.0/(2.0+RHO+SRATIO)))*(SRATIO*FT/SIGYLD)*EPSY
EPSCR4 = EPSY
C = $0.0
FT1 = FT
FT2=(1.0-(1.0+RHO+SRATIO)/10.0)+FT*DEXP((-C+(EPSCR2-EPSCR1)))
FT3 = FT*DEXP((-C*(EPSCR3-EPSCR1))/2.0
FT4 = FT/10.0
C
IF (EPSH .LE. EPSCR2) THEN
SPRING = ((FT1-FT2)+(EPS1-FT1+EPS1))/((EPS1
1 -EPSCR2)+EPS1)
ELSE IF (EPSH .LE. EPSCR2 .AND. EPSH .LE. EPSCR3) THEN
SPRING = ((FT2-FT3)+EPS1+(FT3+EPS3-FT2+EPS3))/((EPS3
1 -EPSCR3)+EPS1)
ELSE IF (EPSH .LE. EPSCR4) THEN
SPRING = ((FT3-FT4)+EPS1+(FT4+EPS4-FT3+EPS4))/((EPS4
1 -EPSCR4)+EPS1)
ENDIF
ENDIF
RETURN
END
C
C*******************************************************************************
C
C INCLUDE(PROCESS)
C
SUBROUTINE COLLINS(EPSMAX, EPSMIN, PROPER, IGAUSS, ILAYER, SIGMOD
$ ,STIFM, FC2MAX, IOUT)
C
IMPLICIT REAL*8 (A-H,0-Z)

C...SWITCHES: REHUMB=100i10, FORHAT=900:10
C...SWITCHES:
C*******************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C*******************************************************************************
C
DIMENSION PROPER(26)

C
EINITL = PROPER(1)
FC = PROPER(2)
EPS = PROPER(7)

C
FC2MAX = 1.0/(0.8+0.34*(EPSMAX/EPS))
IF (FC2MAX .GE. 1.0) FC2MAX = 1.0

C
FC2MAX = FC+FC2MAX
SIGMOD = FC2MAX*(2.0+(DABS(EPSMIN)/EPS)-(DABS(EPSMIN)/EPS)**2)
STIFM = SIGMOD/DABS(EPSMIN)
STIFM = DMAX(EINITL/1000.0, STIFM)

C

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
RETURN
END
C
C************************************************************************
C
C COMPRESSION SOFTENING AFTER CRACKING
C
C INCLUDE(PROCESS)

C SUBROUTINE CHPSFT(EPSMAI, EPSMID, EPSMIH, SIGMA1, SIGMA2, SIGMA3,

C PROPER, EPS1, EPS2, EPS3, L, J, IOUT, CONVER, ESTIFF, CHKGUS, TOLER)

IMPLICIT REAL*8 (A-H,O-Z)
C.. SWITHCES: RENUMB=100:10, FORMAT=900:10
C.. SWITCHES:
C************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C VECCHI
C
C************************************************************************
COMMON/BLOCKS/HISTO(27,70,15), CRACK(27,280,18), HISTY, HIST1
$ , HIST2
COMMON/ABC/THSTF(27,60,18), INOVER, IHSTG1, IHSTG2
DIMENSION PROPER(25)
C REAL*8 HUS
LOGICAL CHKGUS
EMAXOD = HISTO(L,38,J)
EMIDOD = HISTO(L,39,J)
EMINOD = HISTO(L,40,J)
C WRITE(",*) 'MAT STATE IN COMP.SOFT', HISTO(L,29,J)
IF (HISTO(L,29,J) .EQ. 4.0) RETURN
IF (DABS (EPSMAI) .GE. 0.999* DABS (EMAXOD) .OR. DABS (EPSMID) .GE. 0.999*
1 DABS (EMIDOD) .OR. DABS (EPSMIH) .GE. 0.999* DABS (EMINOD) ) THEN
HISTO(L,30,J) = 1.0
HISTO(L,38,J) = EPSMAI
HISTO(L,39,J) = EPSMID
HISTO(L,40,J) = EPSMIH
HISTO(L,41,J) = EPSMAX
HISTO(L,42,J) = EPSmid
HISTO(L,43,J) = EPSMIH
HISTO(L,27,J) = EPS1
HISTO(L,28,J) = EPS2
HISTO(L,29,J) = EPS3
HISTO(L,31,J) = EPS1
HISTO(L,32,J) = EPS2
HISTO(L,33,J) = EPS3
CALL VECCHI (L, J, EPSMAI, EPSMID, EPSMIH, PROPER, EPS1, EPS2
1 , EPS3, SIGMA1, SIGMA2, SIGMA3, CONVER, ESTIFF, CHKGUS, TOLER
RETURN
ELSE
IFLAG3 = 1
PEPSMX = HISTO(L,41,J)
PEPSMD = HISTO(L,42,J)
PEPSMH = HISTO(L,43,J)
IF (DABS (EPSMAI) .LT. DABS (PEPSMX) .AND. DABS (EPSMID) .LT. DABS (PEPSMD)
1 .AND. DABS (EPSMIH) .LT. DABS (PEPSMH) ) THEN
HISTO(L,30,J) = 2.0
VARY1 = EPSMAX/PEPSMX
VARY2 = EPSMID/PEPSMD
VARY3 = EPSMIN/PEPSMH
IF (VARY1 .LT. 0.0 .OR. VARY2 .LT. 0.0 .OR. VARY3 .LT. 0.0 ) THEN
IFLAG3 = 0
RETURN
ELSE
COVER = 999.0
$STIFF = 1
CHKGUS = .TRUE.
IF (VARY1 .LT. 0.0) THEN
HISTO(L,38,J) = 0.0
HISTO(L,41,J) = 0.0
ELSE IF (VARY2 .LT. 0.0) THEN
HISTO(L,39,J) = 0.0
HISTO(L,42,J) = 0.0
ELSE IF (VARY3 .LT. 0.0) THEN
HISTO(L,40,J) = 0.0
HISTO(L,43,J) = 0.0
ENDIF
ELSE
HISTO(L,30,J) = 3.0
ENDIF
IF (IFLAG3 .EQ. 1) THEN
HISTO(L,41,J) = EPSHAX
HISTO(L,42,J) = EPSMID
HISTO(L,43,J) = EPSMH
HISTO(L,31,J) = EPS1
HISTO(L,32,J) = EPS2
HISTO(L,33,J) = EPS3
ENDIF
RETURN
END

C*************************************************************************
C
C INCLUDE(PROCESS)
C
SUBROUTINE VECHEL(IGAUSS,ILAYER,EPSMAX,EPSMID,EPSMH,PROPER,$
$SIGMA1, SIGMA2, SIGMA3, COVER, $STIFF, CHKGUS, TOLER, IOUT)
IMPLICIT REAL*(A-H, O-Z)
C...SWITCHES: REHUMB=100:10, FORMAT=900:10
C...SWITCHES:
C*************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COLLI
C
C*************************************************************************
C
COMMON/BLOCKS/HISTO(27,70,15), CRACK(27,260,15), IHISTY, IHIST1$
$, IHIST2
COMMON/ABC/TESTF(27,80,15), ITSHPG, ITSHSG1, ITSHSG2
DIMENSION PROPER(26)
LOGICAL CHKGUS
C
TOLER = TOLER
EINL = PROPER(1)
IF (HISTO(IGAUSS,30,ILAYER) .EQ. 1.0) THEN
IF (HISTO(IGAUSS,29,ILAYER) .EQ. 1.0) THEN
EPSC = PROPER(7)
IF (DABS(EPSMH) .LE. EPSC) THEN
STATE = 9999
HISTO(IGAUSS,28,ILAYER) = STATE
C
CHECK FAILURE BY CONCRETE REACHING ULTIMATE(COMPRESSIVE).
C
IFLAG = 0
FMAX = HISTO(IGAUSS,68,ILAYER)
CALL COLLI (EPSMAX, EPSMIN, PROPER, IGAUSS, ILAYER, SIGMOD, STIFM, FC2MAX, IOUT)

C
1  C1=100* (HISTO(IGAUSS,20,ILAYER)**2-STIFM**2)/STIFM**2
2  IF (C1.GE. TOLER) THEN
3     HISTO(IGAUSS,20,ILAYER) = (1.0-TOLER/200.)*HISTO(IGAUSS,20,ILAYER)
4     HISTO(IGAUSS,58,ILAYER) = FC2MAX
5     HSTIFF = 1
6     COVER = 999.0
7     WRITE (*, *) 'VEC 1 999'
8     CHKUS = .TRUE.
9     IFLAG = 1
10    ENDIF
11 IF (IFLAG .EQ. 0) THEN
12     STRESS = HISTO(IGAUSS,20,ILAYER)*EPSMIN
13     IF (DABS(STRESS) .GE. DABS(FHAX)) THEN
14         EPSC = PROPER(T)
15         SIGMOD = FMAX^(2.0^(DABS(EPSMIN)/EPSC)-(DABS(EPSMIN)/EPSC)**2)
16         EHEW = DABS(SIGMOD)/DABS(EPSMIN)
17         IF (EHEW .LE. EIHITL/1000.0) EHEW = EIHITL/1000.0
18         HISTO(IGAUSS,20,ILAYER) = EHEW
19         CHKUS = .TRUE.
20         HSTIFF = 1
21         COVER = 999
22         WRITE (*, 20) IFLAG, C1, TOLER,
23         HISTO(IGAUSS,20,ILAYER)
24 ENDIF
25 ENDIF
26 ENDIF
27 IF (DABS(EPSMIN) .GT. DABS(EPS)) THEN
28     HISTO(IGAUSS,29,ILAYER) = 2.0
29     ENEW = HISTO(IGAUSS,29,ILAYER)/DABS(EPS)
30     IF (ENEW .LE. EIHITL/1000.0) ENEW = EIHITL/1000.0
31     HISTO(IGAUSS,20,ILAYER) = ENEW
32     COVER = 999.0
33     WRITE (*, *) 'VEC 2 999'
34     CHKUS = .TRUE.
35     HSTIFF = 1
36 ELSE
37     RETUR
38 ENDIF
39 IF (HISTO(IGAUSS,29,ILAYER) .EQ. 2.0) THEN
40     IF (DABS(SIGMA3) .GE. DABS(HISTO(IGAUSS,68,ILAYER))) THEN
41         HISTO(IGAUSS,29,ILAYER) = 3.0
42         TOLER = 1.0
43         HISTO(IGAUSS,20,ILAYER) = (1.0-TOLER/200.)*HISTO(IGAUSS,20,ILAYER)
44         HISTO(IGAUSS,58,ILAYER) = HISTO(IGAUSS,20,ILAYER)
45         HISTO(IGAUSS,69,ILAYER) = HISTO(IGAUSS,21,ILAYER)
46         HISTO(IGAUSS,34,ILAYER) = SIGMA1
47         HISTO(IGAUSS,35,ILAYER) = SIGMA2
48         HISTO(IGAUSS,36,ILAYER) = HISTO(IGAUSS,68,ILAYER)
49         COVER = 999.0
50         WRITE (*, *) 'VEC 4 999'
51         HSTIFF = 1
52         CHKUS = .TRUE.
53 ELSE
54     ENDIF
55 ENDIF
HISTOCIGAUSS(29, ILAYER) = 1.0
HISTOCIGAUSS(29, ILAYER) = HISTOCIGAUSS(20, ILAYER)
HISTOCIGAUSS(67, ILAYER) = HISTOCIGAUSS(20, ILAYER)
HISTOCIGAUSS(69, ILAYER) = HISTOCIGAUSS(21, ILAYER)
HISTOCIGAUSS(34, ILAYER) = SIGMA1
HISTOCIGAUSS(36, ILAYER) = SIGMA2
HISTOCIGAUSS(36, ILAYER) = HISTOCIGAUSS(26, ILAYER)
COHVER = 999.0
WRITE ('*, *) 'VEC 5 999'
ENDIF
RETURN
ENDIF
IF (HISTOCIGAUSS(29, ILAYER) .EQ. 3.0) THEN
IF (DABS(EPSMIN) .LT. PROPER(8)) THEN
IFLAG = 0
FMAX = HISTOCIGAUSS(36, ILAYER)
CALL COLLIN (EPSMAX, EPSMIN, PROPER, IGAUSS, ILAYER,
SIGMOD, STIFM, FC2MAX, IUUT)
STIFM = DMAX(EHITL/1000.0, STIFM)
CI = 100.0*(HISTOCIGAUSS(20, ILAYER)**2-STIFM**2)/STIFM**2
IF (CI .GE. TOLER) THEN
HISTOCIGAUSS(20, ILAYER) = HISTOCIGAUSS(20, ILAYER)*
(1.0-TOLER/200)
HISTOCIGAUSS(67, ILAYER) = FC2MAX
HISTOCIGAUSS(69, ILAYER) = HISTOCIGAUSS(20, ILAYER)
HISTOCIGAUSS(21, ILAYER) = HISTOCIGAUSS(21, ILAYER)
COHVER = 999.0
WRITE ('*, *) 'VEC 6 999'
HSTIFF = 1
CHKGUS = .TRUE.
IFLAG = 1
ENDIF
ENDIF
IF (IFLAG .EQ. 0) THEN
HISTOCIGAUSS(29, ILAYER) = 4.0
CRACK(IGAUSS, 1, ILAYER) = 0.0
HISTOCIGAUSS(20, ILAYER) = PROPER(1)/1000.0
HISTOCIGAUSS(21, ILAYER) = 0.001
COHVER = 999.0
WRITE ('*, *) 'VECIO 999'
CHKGUS = .TRUE.
HSTIFF = 1
SUBROUTINE VINTAS(GMCRK1,GMCRK2,EPSCRK,SIGMA1,SIGMA2,SIGMA3,PDIR,
                  CRACK,L,J,IOUT,ELNUM,HREL,HLAYRS)
  IMPLICIT REAL*8 (A-H,O-Z)

C...SWITCHES: REAL=8 (A-E,0-2)
C...SWITCHES: INTEGER=8 (A-E,0-2)

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C BISECT FLCHECK LINFO GETHEK GILTX DTRANS
C
C******************************************************************************
INTEGER ELNUM
REAL*8 LTHICK
COMMON/BLKSCR/ESCE(S27,15+ICL+138),CRACK(27,280,I),IHISTY,IHIST1
$       IHIST2
COMMON/ABC/TESTF(S27,15+ICL+138),ITBSGP,ITBSG1,ITBSG2
DIMENSION DGK(6,6),DCS(3,3),LTHICK(9),ZS(9)
REAL*8 HUS,HT(3,3),H(3,3),BSTL(3,3),GCRK(6,6),PDIR(3,3)

C THIS SUBROUTINE IS UTILIZED TO COMPUTE THE SHEAR STIFFNESS OF THE CRACK
C BASED ON EXPERIMENTS BY VIRTEZEOU ET. AL. AND CORRECTING IT WITH THE
C EFFECTS OF REINFORCEMENT AND THE LOCALIZED BOND PRESENT AT THE POINT.
C
ICL = 9*(ICRACK-1) + 1
N(1,1) = CRACK(L,137+ICL,J)
N(1,2) = CRACK(L,138+ICL,J)
N(1,3) = CRACK(L,139+ICL,J)
N(2,1) = CRACK(L,140+ICL,J)
N(2,2) = CRACK(L,141+ICL,J)
N(2,3) = CRACK(L,142+ICL,J)
N(3,1) = CRACK(L,143+ICL,J)
N(3,2) = CRACK(L,144+ICL,J)
N(3,3) = CRACK(L,145+ICL,J)

ICOUNT = 0
SIGNBD = 0.0

DO 120 IM = 1, 3

IF (TESTF(L,IM,J) .EQ. 1.0) THEN
  ISG = TESTF(L,IM+1,J)
  STRES = TESTF(L,18+ISG,J)*TESTF(L,24+ISG,J)
  INMG = (ISG-1)*9 + 1
  NT(1,1) = TESTF(L,30+INMG,J)
  NT(2,1) = TESTF(L,31+INMG,J)
  NT(3,1) = TESTF(L,32+INMG,J)
  NT(1,2) = TESTF(L,33+INMG,J)
  NT(2,2) = TESTF(L,34+INMG,J)
  NT(3,2) = TESTF(L,35+INMG,J)
  NT(1,3) = TESTF(L,36+INMG,J)
  NT(2,3) = TESTF(L,37+INMG,J)
  NT(3,3) = TESTF(L,38+INMG,J)

120 CONTINUE
DO 110 IH = 1, 3
DO 100 IP = 1, 3
BSTL(IH,IP) = 0.0
BSTL(IH,IP) = BSTL(IH,IP) + H(IH,2)*BT(2,IP)
BSTL(IH,IP) = BSTL(IH,IP) + H(IH,3)*BT(3,IP)
100 CONTINUE
110 CONTINUE
STRESS = BSTL(1,1)*BSTL(1,1)*STRES
SIGBBD = SIGBBD + STRESS
ELSE
ICOUBT = ICOUBT + 1
ENDIF
120 CONTINUE
IF (ICOUBT .EQ. 3) SIGBBD = CRACK(L,37+JCRACK, J)
GCRK1 = 0.0
GCRK2 = 0.0
IF (DABS(GMCRK1) .LE. 1.0E-6) GMCRK1 = 1.0E-6
CALL SSSTIF (GMCRK1, SIGBBD, GCRAK1, IQUT, ELNUM, L, J)
130 CONTINUE
IF (DABS(GMCRK2) .LE. 1.0E-6) GMCRK2 = 1.0E-6
CALL SSSTIF (GMCRK2, SIGBBD, GCRAK2, IQUT, ELNUM, L, J)
140 CONTINUE
IF (DABS(GMCRK1) .LE. 1.0E-9 .AND. DABS(GMCRK2) .LE. 1.0E-9) RETURN
D 160 KJM = 1, 6
D 150 KPH = 1, 6
GCRK(KJM, KPH) = 0.0
160 CONTINUE
C THE PERCENTAGE RATIO OF THE CURRENT LAYER IS
C HOW CALCULATED AT CURRENT INTEGRATION POINT BASED ON LAYER TO
C TOTAL THICKNESS.
C
DO 170 JJK = 1, HLAYRS
ROFAC = 0.0
CALL LCHECK (ELNUM, JJK, IFCHK)
1 IF (IFCHK .EQ. 0) THEN
CALL LINFO (JJK, LTHICK, ZS, MATRL, DCS, WNUM, LTHICKL, ZSI, LTHICKL, ZS)
IF (JJK .EQ. J+1) THEN
ROFAC = THICKL/THICKE
ELSE IF (JJK .EQ. J+2) THEN
CALL LCHECK (ELNUM, J + 1, IFCHK)
IF (IFCHK .EQ. 0) ROFAC = THICKL/THICKE
ELSE IF (JJK .EQ. J-1) THEN
ROFAC = THICKL/THICKE
ELSE IF (JJK .EQ. J-2) THEN
CALL LCHECK (ELNUM, J - 1, IFCHK)
IF (IFCHK .EQ. 0) ROFAC = THICKL/THICKE
ELSE
ROFAC = 0.0
ENDIF
ELSE IF (JJK .EQ. J) THEN
ROFAC = 1.0
ELSE
ROFAC = 0.0
ENDIF
GCRK(1,1) = ROFAC*CRACK(L,2,JJK) + GCRK(1,1)
GCRK(1,2) = ROFAC*CRACK(L,3,JJK) + GCRK(1,2)
GCRK(1,3) = ROFAC*CRACK(L,4,JJK) + GCRK(1,3)
GCRK(1,4) = ROFAC*CRACK(L,5,JJK) + GCRK(1,4)
GCRK(1,5) = ROFAC*CRACK(L,6,JJK) + GCRK(1,5)
GCRK(1,6) = ROFAC*CRACK(L,7,JJK) + GCRK(1,6)
GCRK(2,1) = ROFAC*CRACK(L,8,JJK) + GCRK(2,1)
GCRK(2,2) = ROFAC*CRACK(L,9,JJK) + GCRK(2,2)
GCRK(2,3) = ROFAC*CRACK(L,10,JJK) + GCRK(2,3)
GCRK(2,4) = ROFAC*CRACK(L,11,JJK) + GCRK(2,4)
GCRK(2,5) = ROFAC*CRACK(L,12,JJK) + GCRK(2,5)
GCRK(2,6) = ROFAC*CRACK(L,13,JJK) + GCRK(2,6)
GCRK(3,1) = ROFAC*CRACK(L,14,JJK) + GCRK(3,1)
GCRK(3,2) = ROFAC*CRACK(L,15,JJK) + GCRK(3,2)
GCRK(3,3) = ROFAC*CRACK(L,16,JJK) + GCRK(3,3)
GCRK(3,4) = ROFAC*CRACK(L,17,JJK) + GCRK(3,4)
GCRK(3,5) = ROFAC*CRACK(L,18,JJK) + GCRK(3,5)
GCRK(3,6) = ROFAC*CRACK(L,19,JJK) + GCRK(3,6)
GCRK(4,1) = ROFAC*CRACK(L,20,JJK) + GCRK(4,1)
GCRK(4,2) = ROFAC*CRACK(L,21,JJK) + GCRK(4,2)
GCRK(4,3) = ROFAC*CRACK(L,22,JJK) + GCRK(4,3)
GCRK(4,4) = ROFAC*CRACK(L,23,JJK) + GCRK(4,4)
GCRK(4,5) = ROFAC*CRACK(L,24,JJK) + GCRK(4,5)
GCRK(4,6) = ROFAC*CRACK(L,25,JJK) + GCRK(4,6)
GCRK(5,1) = ROFAC*CRACK(L,26,JJK) + GCRK(5,1)
GCRK(5,2) = ROFAC*CRACK(L,27,JJK) + GCRK(5,2)
GCRK(5,3) = ROFAC*CRACK(L,28,JJK) + GCRK(5,3)
GCRK(5,4) = ROFAC*CRACK(L,29,JJK) + GCRK(5,4)
GCRK(5,5) = ROFAC*CRACK(L,30,JJK) + GCRK(5,5)
GCRK(6,1) = ROFAC*CRACK(L,31,JJK) + GCRK(6,1)
GCRK(6,2) = ROFAC*CRACK(L,32,JJK) + GCRK(6,2)
GCRK(6,3) = ROFAC*CRACK(L,33,JJK) + GCRK(6,3)
GCRK(6,4) = ROFAC*CRACK(L,34,JJK) + GCRK(6,4)
GCRK(6,5) = ROFAC*CRACK(L,35,JJK) + GCRK(6,5)
GCRK(6,6) = ROFAC*CRACK(L,36,JJK) + GCRK(6,6)

170 CONTINUE
CALL DTRANS (GCRK, N, IGUT)
ECURBT = GCRK(1,1)
DO 190 KJH = 1, 6
DO 180 KPH = 1, 6
DCOBKJH,KPH) = 0.0
GCRK(KJH,KPH) = 0.0
180 CONTINUE
190 CONTINUE
DO 200 JJK = 1, NLAYRS
ROFAC = 0.0
CALL LCHECK (ELNUM, JJK, IFCHK)
IF (IFCHK .EQ. 0) THEN
    CALL LYEPPO (JJK, LTHICK, ZS, MTH, DCS, NNEL, ELNUM,
                BIPXI, BIPETA, BIPSI)
    CALL GLTLM (L, ELNUM, NNEL, THICKE, RAD)
    IF (JJK .EQ. J+1) THEN
        ROFAC = THICKE/LTHICK
    ELSE IF (JJK .EQ. J+2) THEN
        CALL LCHECK (ELNUM, J + 1, IFCKE)
        IF (IFCKE .EQ. 0) ROFAC = THICKE/LTHICK
        ELSE IF (JJK .EQ. J-1) THEN
        ROFAC = THICKE/LTHICK
    ELSE IF (JJK .EQ. J-2) THEN
        CALL LCHECK (ELNUM, J - 1, IFCKE)
        IF (IFCKE .EQ. 0) ROFAC = THICKE/LTHICK
    ELSE
    END IF
200 CONTINUE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ROFAC = 0.0
ENDIF
ELSE
   ROFAC = 0.0
ENDIF
GCRK(1,1) = ROFAC*CRACK(L2,JJK) + GCRK(1,1)
GCRK(1,2) = ROFAC*CRACK(L3,JJK) + GCRK(1,2)
GCRK(1,3) = ROFAC*CRACK(L4,JJK) + GCRK(1,3)
GCRK(1,4) = ROFAC*CRACK(L5,JJK) + GCRK(1,4)
GCRK(1,5) = ROFAC*CRACK(L6,JJK) + GCRK(1,5)
GCRK(1,6) = ROFAC*CRACK(L7,JJK) + GCRK(1,6)
GCRK(2,1) = ROFAC*CRACK(L8,JJK) + GCRK(2,1)
GCRK(2,2) = ROFAC*CRACK(L9,JJK) + GCRK(2,2)
GCRK(2,3) = ROFAC*CRACK(L10,JJK) + GCRK(2,3)
GCRK(2,4) = ROFAC*CRACK(L11,JJK) + GCRK(2,4)
GCRK(2,5) = ROFAC*CRACK(L12,JJK) + GCRK(2,5)
GCRK(2,6) = ROFAC*CRACK(L13,JJK) + GCRK(2,6)
GCRK(3,1) = ROFAC*CRACK(L14,JJK) + GCRK(3,1)
GCRK(3,2) = ROFAC*CRACK(L15,JJK) + GCRK(3,2)
GCRK(3,3) = ROFAC*CRACK(L16,JJK) + GCRK(3,3)
GCRK(3,4) = ROFAC*CRACK(L17,JJK) + GCRK(3,4)
GCRK(3,5) = ROFAC*CRACK(L18,JJK) + GCRK(3,5)
GCRK(3,6) = ROFAC*CRACK(L19,JJK) + GCRK(3,6)
GCRK(4,1) = ROFAC*CRACK(L20,JJK) + GCRK(4,1)
GCRK(4,2) = ROFAC*CRACK(L21,JJK) + GCRK(4,2)
GCRK(4,3) = ROFAC*CRACK(L22,JJK) + GCRK(4,3)
GCRK(4,4) = ROFAC*CRACK(L23,JJK) + GCRK(4,4)
GCRK(4,5) = ROFAC*CRACK(L24,JJK) + GCRK(4,5)
GCRK(4,6) = ROFAC*CRACK(L25,JJK) + GCRK(4,6)
GCRK(5,1) = ROFAC*CRACK(L26,JJK) + GCRK(5,1)
GCRK(5,2) = ROFAC*CRACK(L27,JJK) + GCRK(5,2)
GCRK(5,3) = ROFAC*CRACK(L28,JJK) + GCRK(5,3)
GCRK(5,4) = ROFAC*CRACK(L29,JJK) + GCRK(5,4)
GCRK(5,5) = ROFAC*CRACK(L30,JJK) + GCRK(5,5)
GCRK(5,6) = ROFAC*CRACK(L31,JJK) + GCRK(5,6)
GCRK(6,1) = ROFAC*CRACK(L32,JJK) + GCRK(6,1)
GCRK(6,2) = ROFAC*CRACK(L33,JJK) + GCRK(6,2)
GCRK(6,3) = ROFAC*CRACK(L34,JJK) + GCRK(6,3)
GCRK(6,4) = ROFAC*CRACK(L35,JJK) + GCRK(6,4)
GCRK(6,5) = ROFAC*CRACK(L36,JJK) + GCRK(6,5)
GCRK(6,6) = ROFAC*CRACK(L37,JJK) + GCRK(6,6)
200 CONTINUE
ES = HISTO(L20,J)
BUS = HISTO(L21,J)
COBST = ES/((1.0+BUS)*(1+BUS))
DCOB(1,1) = COBST*(1.0-BUS)
DCOB(2,2) = COBST*(1.0-BUS)
DCOB(3,3) = COBST*(1.0-BUS)
DCOB(1,2) = COBST+BUS
DCOB(2,1) = DCOB(1,2)
DCOB(3,1) = DCOB(1,2)
DCOB(1,3) = DCOB(1,2)
DCOB(2,3) = DCOB(1,2)
DCOB(3,2) = DCOB(1,2)
DCOB(4,4) = ES/(2.0*(1+BUS))
DCOB(5,5) = DCOB(4,4)
DCOB(6,6) = DCOB(4,4)
DO 220 KKP = 1, 6
   DO 210 KJJ = 1, 6
      GCRK(KKP,KJJ) = GCRK(KKP,KJJ) + DCOB(KKP,KJJ)
   CONTINUE
210 CONTINUE
220 CONTINUE
CALL DTRANS (GCRK, N, IOUT)
IF (DABS(GMCR1) .GT. 1.E-7) GCRAK1 = GCRAK1*ECURBT/GCRK(1,1)
IF (DABS(GMCR2) .GT. 1.E-7) GCRAK2 = GCRAK2*ECURNT/GCRK(1,1)
GFLR = HISTO(L,26,J)/(2.0*(1.0+HISTO(L,27,J)))
IF (GCRAK1 .LE. 0.01*GFLR) GCRAK1 = 0.010*GFLR
IF (GCRAK2 .LE. 0.01+GFLR) GCRAK2 = 0.010*GFLR
GCRAK1 = DMIN1(10.0*GFLR,GCRAK1)
GCRAK2 = DMIN1(10.0*GFLR,GCRAK2)
GCRAK1 = DMIN1(CRACK(L,97+ICRACK,J),GCRAK1)
GCRAK2 = DMIN1(CRACK(L,107+ICRACK,J),GCRAK2)
CRACK(L,97+ICRACK,J) = GCRAK1
CRACK(L,107+ICRACK,J) = GCRAK2
RETURN
END

C*************************************** *************
INTEGER ELNUM
C
COMMON/LPROP/PROPER(25,16)
COMMON/LAYERA/ELLYFO(320,5000)
C
SUBROUTINE RETURNS A FLAG TO IDENTIFY GIVEN LAYER AS STEEL/OTHERS.
C
KK = 21*(ILAYER-1) + 1
MATRL = ELLYFO(KK,ELNUM)
IFCHK = PROPER(25,MATRL)
C
RETURN
END

C*************************************** *************
INTEGER ELNUN
DIMENSION A(20),B(20),C(20),D(20),E(20)
C
C
A -- SHEAR STRAIN DATA
B -- SHEAR STRESS WITH SIGNORMAL=0.0 (DOWEL TESTS)
C -- SHEAR STRESS WITH SIGNORMAL=0.6MPA (AGG. INTERLOCK)
D -- SHEAR STRESS WITH SIGNORMAL=1.0MPA (AGG. INTERLOCK)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
E -- Shear Stress with Sigma Normal=2.0 MPa (AGG. Interlock)

\[
\begin{align*}
A(1) &= 0.0 \\
A(2) &= 4.16E-4 \\
A(3) &= 8.33E-4 \\
A(4) &= 1.25E-3 \\
A(5) &= 1.66E-3 \\
A(6) &= 2.00E-3 \\
A(7) &= 3.33E-3 \\
A(8) &= 4.16E-3 \\
A(9) &= 5.00E-3 \\
A(10) &= 6.03E-3 \\
A(11) &= 6.66E-3 \\
A(12) &= 7.60E-3 \\
A(13) &= 8.33E-3 \\
A(14) &= 9.16E-3 \\
A(15) &= 9.99E-3 \\
A(16) &= 0.01083 \\
A(17) &= 0.01166 \\
A(18) &= 0.01250 \\
A(19) &= 0.01333 \\
B(1) &= 0.0 \\
B(2) &= 0.644 \\
B(3) &= 0.777 \\
B(4) &= 0.833 \\
B(5) &= 0.888 \\
B(6) &= 0.976 \\
B(7) &= 1.035 \\
B(8) &= 1.133 \\
B(9) &= 1.200 \\
B(10) &= 1.255 \\
B(11) &= 1.288 \\
B(12) &= 1.334 \\
B(13) &= 1.388 \\
B(14) &= 1.422 \\
B(15) &= 1.433 \\
B(16) &= 1.444 \\
B(17) &= 1.466 \\
B(18) &= 1.488 \\
B(19) &= 1.500 \\
C(1) &= 0.0 \\
C(2) &= 1.500 \\
C(3) &= 1.600 \\
C(4) &= 1.670 \\
C(5) &= 1.750 \\
C(6) &= 1.680 \\
C(7) &= 2.000 \\
C(8) &= 2.100 \\
C(9) &= 2.250 \\
C(10) &= 2.350 \\
C(11) &= 2.400 \\
C(12) &= 2.450 \\
C(13) &= 2.500 \\
C(14) &= 2.500 \\
C(15) &= 2.500 \\
C(16) &= 2.500 \\
C(17) &= 2.500 \\
C(18) &= 2.500 \\
C(19) &= 2.500 \\
D(1) &= 0.0 \\
D(2) &= 2.000 \\
D(3) &= 2.500 \\
D(4) &= 2.750
\end{align*}
\]
D(6) = 2.80
D(7) = 3.15
D(8) = 3.30
D(9) = 3.50
D(10) = 3.65
D(11) = 3.60
D(12) = 3.70
D(13) = 3.75
D(14) = 3.80
D(15) = 3.85
D(16) = 3.87
D(17) = 3.90
D(18) = 3.95
D(19) = 4.00
E(1) = 0.0
E(2) = 2.00
E(3) = 2.85
E(4) = 3.25
E(5) = 3.60
E(6) = 4.60
E(7) = 5.0
E(8) = 5.30
E(9) = 5.60
E(10) = 5.85
E(11) = 5.80
E(12) = 5.75
E(13) = 5.70
E(14) = 5.60
E(15) = 5.50
E(16) = 5.40
E(17) = 5.25
E(18) = 5.10
E(19) = 5.00
TAUCRACK = 0.0
GCRAK = 0.0
GAM1 = 0.0
GAM2 = 0.0
TAU1 = 0.0
TAU2 = 0.0
TAU3 = 0.0
TAU4 = 0.0
TAU5 = 0.0
TAU6 = 0.0
TAU7 = 0.0
TAU8 = 0.0
TAU12 = 0.0
TAU34 = 0.0
TAU56 = 0.0
TAU78 = 0.0
FX0 = 0.0
FX1 = 0.0
FX2 = 0.0
FX3 = 0.0
FX4 = 0.0
FX5 = 0.0
FX6 = 0.0
FX7 = 0.0
FX8 = 0.0
FX9 = 0.0
FX10 = 0.0
FX11 = 0.0
FX12 = 0.0
FX13 = 0.0
FX14 = 0.0
FX15 = 0.0
FX16 = 0.0
FX17 = 0.0
FX18 = 0.0
FX19 = 0.0

C FOR CURRENT STRAIN FIND THE UPPER AND LOWER EIPT VALUES
C OF THE STRAIN AND THE CORRESP. STRESSES, FOR EACH NORMAL LOAD CURVE GIVEN HERE. FIND STRESS VALUES FOR CURRENT STRAIN ON EACH OF THESE CURVES (LINEAR INTERPOLATE) AND SUBSEQUENTLY INTERPOLATE THESE VALUES OVER THE CURRENT NORMAL STRESSES.
360

USING DEVIDED DIFFERENCES.

INDEX = 0
DO 100 I = 1, 18
  IF (DABS(GAMCRK) .GE. A(I)) .AND. DABS(GAMCRK).LT.A(I+1)) THEN
    INDEX = I
    GO TO 110
  ENDIF
100 CONTINUE

IF (DABS(GAMCRK) .GE. A(19)) THEN
  TAU12C = B(19)
  TAU36C = C(19)
  TAU58C = D(19)
  TAU78C = E(19)
  GO TO 120
ENDIF

110 CONTINUE

GAM1 = A(INDEX)
GAM2 = A(INDEX+1)
TAU1 = B(INDEX)
TAU2 = B(INDEX+1)
TAU3 = C(INDEX)
TAU4 = C(INDEX+1)
TAU5 = D(INDEX)
TAU6 = D(INDEX+1)
TAU7 = E(INDEX)
TAU8 = E(INDEX+1)

TAU12C = TAU1 + (TAU2-TAU1)/(GAM2-GAM1)*DABS(GAMCRK)-GAM1)
TAU36C = TAU3 + (TAU4-TAU3)/(GAM2-GAM1)*DABS(GAMCRK)-GAM1)
TAU56C = TAU5 + (TAU6-TAU5)/(GAM2-GAM1)*DABS(GAMCRK)-GAM1)
TAU78C = TAU7 + (TAU8-TAU7)/(GAM2-GAM1)*DABS(GAMCRK)-GAM1)

USING DEVIDED DIFFERENCES FIND CORRECT TAU FOR CURRENT SIGNORMAL

120 CONTINUE

FX0 = TAU12C
FX01 = (TAU36C-TAU12C)/0.50
FX12 = (TAU56C-TAU36C)/0.50
FX34 = (TAU78C-TAU56C)/1.0
FX012 = FX12 - FX01
FX123 = (FX34-FX12)/1.5
FX0123 = (FX123-FX012)/2.0

TAUCRK = FX0 + FX01*SIGBND + FX012*(SIGBND-0.50)*SIGBND + FX0123*(

1 SIGBND-1.0)*SIGBND+FX0123*(

C CRACK SHEAR STIFFNESS IS ...

GCRACK = DABS(TAUCRK/GAMCRK)

RETURN

ED0
C*************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DMATCS CSSSN
C*************************************************************************

INTEGER ELNUM
C
IF (ICODE .EQ. 0) THEN
   IPG = 0
   CALL DMATCS (ELNUM, ITYPE, INTGPN, IFLAG, ICODE, IPG)
ELSE
   CALL CSSSN (ELNUM, ITYPE, INTGPN, IFLAG, IDUT, HNEL, ICODE)
ENDIF
RETURN
END

C*************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C*************************************************************************

CHARACTER*105 DUMMY
REAL*8 LTHICK
INTEGER ELNUM
COMMON/UTIL1/STRESS(6),STRAIN(6),DUMMY
COMMON/MATER1/DEP(6,6)
COMMON/ELSTRI/STAR(6)
COMMON/ELSTRI/STAR(6)
COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDEV6,LDEVICE,LDEVST
COMMON/CNTROL/INCREM,NDQ
COMMON/IALGTH/IFLAGE
COMMON/LAYERS/LTHICK(9),ZS(9),DCS(3,3),NLAYRS,MATRL,LYNUM
COMMON/LPROP/PROPER(25,15)
DIMENSION DE(6),DS(6)
C
IF (ITYPE .GT. 300) THEN
   IEDE = 6
ELSE
   IEDE = 4
ENDIF
C
IF (INCREM.GT.1 .OR. NDQ.GT.1) THEN
   CALL IDGET (LDEV1, 96, 'LDEV1', 6)
ELSE
   DO 100 K1 = 1, IEDE
      STRESS(K1) = 0.
      STRAIN(K1) = 0.
   100 CONTINUE
ENDIF
C
IF (IFLAGE .EQ. 1) THEN
   DO 110 K1 = 1, IEDE
      DUMMY
ENDIF

C*************************************************************************
C*************************************************************************
C*************************************************************************
C*************************************************************************
DE(K1) = STRH(K1) - STRAIH(K1)

ELSE IF (IFLAG5 .EQ. 0) THEN
   DO 120 K1 = 1, IEDD
   DECK1) = STRBCK1)
120 CONTINUE

IFG = 0
CALL DMATCS (ELNUM, ITYPE, IHTGPB, IFLAG, IOUT, ICODE, IPG)

DO 140 K1 = 1, IEDD
   S = 0.
   DO 130 K2 = 1, IEDD
      S = S + DEP(K1,K2)*DE(K2)
130 CONTINUE
   DS(K1) = S
140 CONTINUE

IF (IFLAGB .EQ. 1) THEN
   STRESS(K1) = STRESS(K1) + DS(K1)
ELSE IF (IFLAGB .EQ. 0) THEN
   STRESS(K1) = DS(K1)
ENDIF

STRS(K1) = STRESS(K1)

IF (IFLAGB .EQ. 1) THEN
   CALL IOPUT (LDEV2, 96, '( A 96 )', 5)
ELSE IF (IFLAGB .EQ. 0) THEN
   CALL IOPUT (LDEV3, 96, '( A 96 )', 5)
ENDIF
RETURN

900 FORMAT(2A48)
END

C
C ***------------------------------------------ D M A T C S ------------------------------------------
C
C INCLUDE (PROCESS)
C SUBROUTINE DMATCS (ELNUM, ITYPE, IHTGPB, IFLAG, IOUT, ICODE, IPG)
C
C ***------------------------------------------
C I
C PROGRAM 'DMATCS' EVALUATES THE STRESS-STRAIN STIFFNESS MATRIX
C FOR NONLINEAR INELASTIC MATERIALS
C I
C COMMON BLOCKS
C I
C ***------------------------------------------
C
C IMPLICIT REAL*8 (A-H, O-Z)
C
C...SWITCHES: RENUM=100:10,FORMAT=900:10
C...SWITCHES:
C***------------------------------------------
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C DSHELL
C
C***------------------------------------------
C REAL*8 LTHICK

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
integer elhum
common/lprop/proper(35,15)
common/mater1/dep(6,6)
common/layerl/layer(9),z(9),dcs(3,3),elayers,matal,lyhum
common/blockl/histo(27,70,15),crack(27,250,15),ihisty,ihistl,ihist2

if (itype .gt. 300) then
  do 110 k2 = 1, 6
      do 100 k1 = 1, 6
          dep(k1,k2) = 0.
    continue
100    continue
110    continue

  dep(1,1) = crack(intop,2,lyhum)
  dep(1,2) = crack(intop,3,lyhum)
  dep(1,3) = crack(intop,4,lyhum)
  dep(1,4) = crack(intop,5,lyhum)
  dep(1,5) = crack(intop,6,lyhum)
  dep(1,6) = crack(intop,7,lyhum)
  dep(2,1) = crack(intop,8,lyhum)
  dep(2,2) = crack(intop,9,lyhum)
  dep(2,3) = crack(intop,10,lyhum)
  dep(2,4) = crack(intop,11,lyhum)
  dep(2,5) = crack(intop,12,lyhum)
  dep(2,6) = crack(intop,13,lyhum)
  dep(3,1) = crack(intop,14,lyhum)
  dep(3,2) = crack(intop,15,lyhum)
  dep(3,3) = crack(intop,16,lyhum)
  dep(3,4) = crack(intop,17,lyhum)
  dep(3,5) = crack(intop,18,lyhum)
  dep(3,6) = crack(intop,19,lyhum)
  dep(4,1) = crack(intop,20,lyhum)
  dep(4,2) = crack(intop,21,lyhum)
  dep(4,3) = crack(intop,22,lyhum)
  dep(4,4) = crack(intop,23,lyhum)
  dep(4,5) = crack(intop,24,lyhum)
  dep(4,6) = crack(intop,25,lyhum)
  dep(5,1) = crack(intop,26,lyhum)
  dep(5,2) = crack(intop,27,lyhum)
  dep(5,3) = crack(intop,28,lyhum)
  dep(5,4) = crack(intop,29,lyhum)
  dep(5,5) = crack(intop,30,lyhum)
  dep(5,6) = crack(intop,31,lyhum)
  dep(6,1) = crack(intop,32,lyhum)
  dep(6,2) = crack(intop,33,lyhum)
  dep(6,3) = crack(intop,34,lyhum)
  dep(6,4) = crack(intop,35,lyhum)
  dep(6,5) = crack(intop,36,lyhum)
  dep(6,6) = crack(intop,37,lyhum)
endif

if (itype .eq. 4) then
  if (proper(28,matal) .ne. 0.0) call dshell (isg, icode, elhum, intop, ifg)
endif
endif
RETURN
END

C======================================================================================================
C INCLUDE (PROCESS)
C SUBROUTINE DHELL (IOUT,ICODE,ELNUM,INTGPH,IPG)
C======================================================================================================
C I
C I  PROGRAM 'DMATCS' EVALUATES THE STRESS-STRAIN STIFFNESS MATRIX
C I FOR NONLINEAR INELASTIC MATERIALS
C I
C I COMMON B LOCKS
C I
C======================================================================================================
C IMPLICIT REAL*8 (A-E,O-Z)
C ..SWITCHES: REHUME=100:10,FORMAT=900:10
C ..SWITCHES:
C======================================================================================================
C C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C SIMUL
C C SIMUL
C======================================================================================================
C INTEGER ELNUM
C REAL*8 LTHICK
C COMMON/MATER1/DEP(6,6)
C COMMON/TRANS/V1(3),V2(3),V3(3)
C COMMON/LAYER/LTHICK(9),ZS(9),DCS(3,3),NLYERS,MATRL,LYNUM
C COMMON/INPUT/BIPXI,BIPETA,BIPSI,HIP,IBTCOD
C COMMON/TSHHEAR/AK1(5000,9),AK2(5000,9),AF1(5000,15,27),AF2(5000,15,27)
C DIMEDSIOH TEMPOS,6) ,D C(6,6) ,TMP3(6,6)
C
C IP0INT = INTGPH* (BIPXI*HIPETA)
C INDEX = INTGPH - IP0INT+HIPXI*HIPETA
C IF (INDEX .EQ. 0) INDEX = HIPXI*HIPETA
C
C DC(1,1) = V1(1)**2
C DC(2,1) = V2(1)**2
C DC(3,1) = V3(1)**2
C DC(1,2) = V1(2)**2
C DC(2,2) = V2(2)**2
C DC(3,2) = V3(2)**2
C DC(1,3) = V1(3)**2
C DC(2,3) = V2(3)**2
C DC(3,3) = V3(3)**2
C DC(4,1) = 2.*V1(1)*V2(1)
C DC(6,1) = 2.*V2(1)*V3(1)
C DC(6,1) = 2.*V1(1)*V3(1)
C DC(4,2) = 2.*V1(2)*V2(2)
C DC(6,2) = 2.*V2(2)*V3(2)
C DC(6,2) = 2.*V1(2)*V3(2)
C DC(4,3) = 2.*V1(3)*V2(3)
C DC(6,3) = 2.*V2(3)*V3(3)
C DC(6,3) = 2.*V1(3)*V3(3)
C DC(1,4) = V1(1)*V1(2)
C DC(1,4) = V1(1)*V1(2)
C DC(2,4) = V2(1)*V2(2)
C DC(3,4) = V3(1)*V3(2)
C DC(1,5) = V1(2)*V1(3)
C DC(2,5) = V2(2)*V2(3)
\[ DC(3,5) = V_3(2) \cdot V_3(3) \]

\[ DC(1,6) = V_1(1) \cdot V_1(3) \]

\[ DC(2,6) = V_2(1) \cdot V_2(3) \]

\[ DC(3,6) = V_3(1) \cdot V_3(3) \]

\[ DC(4,4) = V_1(1) \cdot V_2(2) + V_2(1) \cdot V_1(2) \]

\[ DC(4,6) = V_1(1) \cdot V_1(2) + V_1(2) \cdot V_1(1) \]

\[ DC(6,4) = V_3(1) \cdot V_3(2) + V_3(2) \cdot V_3(1) \]

\[ DC(6,6) = V_3(1) \cdot V_1(2) + V_1(1) \cdot V_3(2) \]

\[ DC(4,6) = V_1(3) \cdot V_2(2) + V_2(3) \cdot V_1(2) \]

\[ DC(6,6) = V_3(3) \cdot V_1(2) + V_1(3) \cdot V_3(2) \]

\[ DC(6,6) = V_3(3) \cdot V_3(1) + V_3(1) \cdot V_3(3) \]

C ---- TRANSFORM DEP FROM GLOBAL TO LOCAL COORDINATES BY USING THE
C transformation matrix DC.

C

DO 110 K1 = 1, 6
DO 100 K2 = 1, 6
TEMP3(K1,K2) = DC(K1,K2)
100 CONTINUE
110 CONTINUE

N = 6
EPS = 0.0001
NR = 6
DETER = SIMUL(N,TEMP3,_EPS,NR,4)
IF (DETER .EQ. 0.0) THEN
WRITE (* , 3) 'ERROR IN SUB ELASTIC---SIMUL INVERSE'
STOP
ENDIF
C

DO 140 K1 = 1, 6
DO 130 K2 = 1, 6
TEMP1 = 0.
DO 120 K3 = 1, 6
TEMP1 = TEMP1 + TEMP3(K3,K1)*DEP(K3,K2)
120 CONTINUE
130 CONTINUE
140 CONTINUE
C

DO 170 K1 = 1, 6
DO 160 K2 = 1, 6
TEMP1 = 0.
DO 150 K3 = 1, 6
TEMP1 = TEMP1 + TEMP(K1,K3)*TEMP3(K3,K2)
150 CONTINUE
160 CONTINUE
170 CONTINUE
C

---- The shell normal stiffness(local) is suppressed here
C

DO 210 I = 1, 6
IF (I .EQ. 3) GO TO 200
DO 190 J = 1, 6
IF (J .EQ. 3) GO TO 180
DEP(I,J) = DEP(I,J) - DEP(3,J)*DEP(1,3)/DEP(3,3)
180 CONTINUE
190 CONTINUE
200 CONTINUE
210 CONTINUE
DO K1 = 1, 6
DEP(3,1) = 0.0
DEP(1,3) = 0.0
END DO
C
C ... SHEAR CORRECTION FACTOR FOR SHELLS
C
AFACT2 = AF1(ELSUM,LSUM,INTGP)*5./6.
AFACT1 = AF2(ELSUM,LSUM,INTGP)*5./6.
C
IF (NLAYS .EQ. 1) THEN
AFACT1 = 5./6.
AFACT2 = 5./6.
ENDIF
C
DEP(6,6) = DEP(6,6)*AFACT1
DEP(6,6) = DEP(6,6)*AFACT2
C
IF (IPG .EQ. 1) RETURN
C
C ------ TRANSFORM DEP TO THE GLOBAL COORDINATES BY USING THE
C transformation matrix <DC.
C
DO 260 K1 = 1, 6
DO 240 K2 = 1, 6
    TEMP1 = 0.
    DO 230 K3 = 1, 6
        TEMP1 = TEMP1 + DC(K3,K1)*DEP(K3,K2)
    230 CONTINUE
    TEMPI = TEMP1
    DEP(K1,K2) = TEMPI
240 CONTINUE
260 CONTINUE
C
DO 260 K1 = 1, 6
DO 270 K2 = 1, 6
    TEMP1 = 0.
    DO 260 K3 = 1, 6
        TEMP1 = TEMP1 - TEMPI + DC(K3,K1)*DEP(K3,K2)
    260 CONTINUE
    TEMPI = TEMP1
    DEP(K1,K2) = TEMPI
270 CONTINUE
280 CONTINUE
C
RETURN
END

C
C =============== GAUSS ===============
C
C SUBROUTINE GAUSS(HIP,W,GCOORD)
C
C ==================================================================
C I
C I SUBPROGRAM GAUSS STORES THE COORDINATES XI AND ETA OF THE
C I FOURTEEN INTEGRATION POINTS AND THEIR WEIGHTING FUNCTIONS
C FOR THE FOUR POINT AND THE EIGHT POINT INTEGRATION.
C I
C I HIP = NUMBER OF THE INTEGRATION POINTS
C I W(I) = WEIGHT FUNCTION
C I GCOORD(I) = COORDINATES OF THE GAUSSIAN POINTS
C I FROM THE NEGATIVE TO POSITIVE
C I
C ===============
C
C Subroutine: GAUSS
C ...SWITCHES: RENUM=100:10,FORMAT=900:10
C ...SWITCHES:
C******************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C******************************************************************************
REAL*8 W,GCOORD
DIMENSION W(4),GCOORD(4)
C IF (HIP .EQ. 1) THEN
  W(1) = 2.0DO
  GCOORD(1) = 0.0DO
C ELSE IF (HIP .EQ. 2) THEN
  W(1) = 1.0DO
  W(2) = 1.0DO
  GCOORD(1) = -0.777350269189626D0
  GCOORD(2) = 0.777350269189626D0
C ELSE IF (HIP .EQ. 3) THEN
  W(1) = 5.0DO/9.0DO
  W(2) = 8.0DO/9.0DO
  W(3) = W(1)
  GCOORD(1) = -0.774596669241483D0
  GCOORD(2) = 0.
  GCOORD(3) = 0.774596669241483D0
C ELSE IF (HIP .EQ. 4) THEN
  W(1) = 0.347854845137484DO
  W(2) = 0.652145154862566DO
  W(3) = W(2)
  W(4) = W(1)
  GCOORD(1) = -0.861136311594053DO
  GCOORD(2) = -0.339981043584886DO
  GCOORD(3) = 0.339981043584886DO
  GCOORD(4) = 0.861136311594053DO
C ELSE
  GO TO 100
C ENDIF
100 CONTINUE
RETURN
END
C******************************************************************************
C INCLUDE (PROCESS)
SUBROUTINE ISHAPE
******************************************************************************
C THIS PROGRAM EVALUATES THE SHAPE FUNCTIONS, THEIR DERIVATIVES
C WITH RESPECT TO THE NATURAL COORDINATES, AND THE WEIGHT
C FUNCTIONS AT EACH INTEGRATION POINT.
C ENTRY POINTS:
C Ish2DG (FOR 2D ELEMENTS)
C Ish3DG (FOR 3D ELEMENTS)
C I
C =============================================================
C IMPLICIT REAL*8 (A-H.O-Z)
C...SWITCHES: REUMR=100:10,FORMAT=900:10
C...SWITCHES:
C-----------------------------------------------------------------
C C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C GAUSS N2D ISOP2D ISHEXT ISOP3D
C GAUSSL
C C-----------------------------------------------------------------
REAL*8 N,NXI,NETA,NSI,F,FXI,FETA,FSI,ASI,SI,LTHICK
COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27),SI(27)
COMMON/ISHAP2/W(27)
COMMON/CONSIS/NSTLOC(27,3)
COMMON/INPUT1/HIPXI,HIPETA,HIPS1,NIP,INTCOD
COMMON/LAYER/LTHICK(9),ZS(9),DCS(3,3),BLAYS,MSTL,LYSUM
DIMENSION F(20),FXI(20),FETA(20),FSI(20)
DIMENSION WXI(4),WETA(4),WSI(4),XI(4),ETA(4),SIP(4)
C C EVALUATE THE SHAPE FUNCTIONS AND THEIR DERIVATIVES WITH RESPECT
C TO THE ISOPARAMETRIC COORDINATES FOR TWO DIMENSIONAL ELEMENTS
C AT EACH GAUSSIAN INTEGRATION POINT.
C ENTRY ISH2DG(ITYPE, NNEL, IERROR)
C C ***** GET THE ISOPARAMETRIC COORDINATES OF INTEGRATION POINTS
C CALL GAUSS (HIPXI, WXI, XI)
CALL GAUSS (HIPETA, WETA, ETA)
NIP = NIPXI*NIPETA
C C ***** HIP = TOTAL NUMBER OF INTEGRATION POINTS FOR THE ELEMENT
C ***** IETA = ROW NUMBER OF THE GAUSSIAN POINT FROM THE BOTTOM
C ***** IXI = COLUMN NUMBER OF THE GAUSSIAN POINT FROM LEFT
C DO 110 IETA = 1, NIPETA
DO 100 IXI = 1, NIPXI
J = (IETA-1)*NIPXI + IXI
W(J) = WXI(IXI)*WETA(IETA)
AXI = XKIXI)
AETA = ETA(IETA)
CALL N2D (AXI, AETA, F, ITYPE, TERROR)
CALL IS0P2D (AXI, AETA, FXI, FETA, ITYPE, TERROR)
CALL ISHEXT (NNEL, J, F, FXI, FETA, FSI, ASI)
100 CONTINUE
110 CONTINUE
C C C C C C
C ------- EVALUATE THE SHAPE FUNCTIONS AND THEIR DERIVATIVES WITH RESPECT
C TO THE ISOPARAMETRIC COORDINATES FOR THREE DIMENSIONAL ELEMENTS
C AT EACH GAUSSIAN INTEGRATION POINT.
C ENTRY ISH3DG (ITYPE, NHEL, IERROR)
CALL GAUSS (BIPXI, WIXI, XI)
CALL GAUSS (BIPETA, WETA, ETA)
CALL GAUSS (BIPSI, WSI, SIP)
C WRITE(6,'*') 'ISHS030 ENTERED'
NIP = BIPXI*BIPETA*BIPSI
I = BIPXI*WIPETA
C DO 140 ISI = 1, NIP
   DO 130 IETA = 1, NIPSI
      DO 120 IXI = 1, NIPXI
         J = (ISI-1)*I + (IETA-1)*BIPXI + IXI
         W(J) = WIXI(IXI)*WETA(IETA)*WSI(ISI)
         AXI = XKIXI
         ETA = ETA(IETA)
         PSI = SIP(ISI)
C*****
      RXSTLOC(J,1) = AXI
      RXSTLOC(J,2) = ETA
      RXSTLOC(J,3) = PSI
C*****
   C
   120 CONTINUE
   130 CONTINUE
   140 CONTINUE
C RETURN
C ----- EVALUATE THE SHAPE FUNCTIONS AND THEIR DERIVATIVES WITH RESPECT
C TO THE ISOPARAMETRIC COORDINATES FOR THREE DIMENSIONAL SHELL
C ELEMENTS AT EACH GAUSSIAN INTEGRATION POINT.
C
ENTRY ISHSHL (ITYPE, NHEL, TERROR)
C IT = ITYPE - 100
CALL GAUSS (BIPXI, WIXI, XI)
CALL GAUSS (BIPETA, WETA, ETA)
IF (HLAYRS .EQ. 1) CALL GAUSS (BIPSI, WSI, SIP)
NIP = BIPXI*BIPETA*BIPSI
I = BIPXI*WIPETA
C
   DO 170 K1 = 1, 20
      FSI(K1) = 0.000
   170 CONTINUE

C DO 200 ISI = 1, NIP
   DO 190 IETA = 1, NIPSI
      DO 180 IXI = 1, NIPXI
         J = (ISI-1)*I + (IETA-1)*BIPXI + IXI
         IF (HLAYRS .EQ. 1) THEN
            W(J) = WIXI(IXI)*WETA(IETA)*WSI(ISI)
            AXI = XI(IXI)
            ETA = ETA(IETA)
            PSI = SIP(ISI)
            ENDF1
            AXI = XI(IXI)
            ETA = ETA(IETA)
         CALL B2D (AXI, ETA, F, IT, TERROR)
         CALL ISOP2D (AXI, ETA, FXI, FETA, ITYPE, TERROR)
         IF (HLAYRS .GT. 1) THEN
            CALL GAUSSL (F, HIPDS, WIPSI, BIPSI, 1ST, BIPSI)
         ENDIF
      ENDIF
   180 CONTINUE
   190 CONTINUE
   200 CONTINUE
ASI = SIP(ISI)
W(J) = WI(IIX)*WETA(IETA)*WSI(ISI)
ENDIF
RSTLOC(J,1) = AXI
RSTLOC(J,2) = AETA
RSTLOC(J,3) = AXI
CALL ISHEXT (WHEL, J, F, FII, FETA, FSI, ASI)
180 CONTINUE
190 CONTINUE
200 CONTINUE
C
RETURN
END
C
******************************************************************************
C INCLUDE(PROCESS)
SUBROUTINE GAUSSLCF.SIP.WSI.HHEL.ISI,HIPSI)
C
IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C
....SWITCHES; REHUB=100:10; FORHAT=900:10
C....SWITCHES;
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C GETLTK
C
REAL*8 LTKISI
DIMENSION F(20),WSI(4),SIP(4)
CALL GETLTK (WHEL, THICKE, ZSI, F, THICKL)
C
ZIISI = ZSI*2.0/THICKE
LTKISI = THICKL*2.0/THICKE
IF (HIPSI .EQ. 1) THEN
  IF (ISI .EQ. 1) THEN
    SIP(ISI) = ZIISI
    WSI(ISI) = 2.0*THICKL/THICKE
  ENDIF
ELSE IF (HIPSI .EQ. 2) THEN
  IF (ISI .EQ. 1) THEN
    SIP(ISI) = ZIISI - (LTKISI/2.0)*0.57736
    WSI(ISI) = THICKL/THICKE
  ELSE IF (ISI .EQ. 2) THEN
    SIP(ISI) = ZIISI + (LTKISI/2.0)*0.57736
    WSI(ISI) = THICKL/THICKE
  ENDIF
ELSE IF (HIPSI .EQ. 3) THEN
  IF (ISI .EQ. 1) THEN
    SIP(ISI) = ZIISI - (LTKISI/2.0)*0.77469669241483
    WSI(ISI) = (THICKL/THICKE)*5.0/9.0
  ELSE IF (ISI .EQ. 2) THEN
    SIP(ISI) = ZIISI
    WSI(ISI) = (THICKL/THICKE)*8.0/9.0
  ELSE IF (ISI .EQ. 3) THEN
    SIP(ISI) = ZIISI + (LTKISI/2.0)*0.77469669241483
    WSI(ISI) = (THICKL/THICKE)*5.0/9.0
  ENDIF
ENDIF
ENDIF
RETURN
END
C ======================= GET L A Y E R T H I C K ======================
C INCLUDE(P R O C E S S)
SUBROUTINE GETLTK(HEL,THICK,ZS,F,THICKL)
C IMPLICIT DOUBLE PRECISION(A-H,O-Z)
C ...SWITCHES: RESUME=100:10,FORMAT=900:10
C ...SWITCHES:
C***************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C***************************************************************************
C DO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C***************************************************************************
REAL*4 THICK
REAL*8 LTHICK
COMMOD/LAYERS/LTHICK(9),ZS(9),DCS(3,3),N L A Y R S,M A T R L,LYHUM
COMMOD/INPUT9/THICK(9),IFLAG
DIMENSION F(20)
C THICK = 0.0
THICKL = 0.0
ZSI = 0.0
DO 100 K = 1, HEL
ZSI = ZSI + F(K)*ZS(K)
THICKL = THICKL + LTHXXCK(K)*F(K)
THICKE = THICKE + THICK(K)*F(K)
100 CONTINUE
RETURN
END
C =========================== I S H E X T  ====================
C INCLUDE(P R O C E S S)
SUBROUTINE ISHEXT(HEL,J,F,FXI,FETA,FSI,ASI)
C ...SWITCHES: RESUME=100:10,FORMAT=900:10
C ...SWITCHES:
C***************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C***************************************************************************
C DO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C***************************************************************************
REAL*8 E,HXI,BETA,NSI,F,FXI,FETA,FSI,ASI,SI
COMMOD/ISHAP1/H(20,27),HXI(20,27),BETA(20,27),HSI(20,27),SI(27)
DIMENSION F(20),FXI(20),FETA(20),FSI(20)
C VDIR: ASSUME CDUNT(8)
C DO 100 HH = 1, HEL
B(HH,J) = F(HH)
HXI(HH,J) = FXI(HH)
HETA(HH,J) = FETA(HH)
HSI(HH,J) = FSI(HH)
SI(J) = ASI
100 CONTINUE
C VDIR: ASSUME CDUNT(8)
C DO 99 H = 1, HEL
E(H,J) = F(H)
FXI(H,J) = FXI(H)
BETA(H,J) = FETA(H)
HSI(H,J) = FSI(H)
SI(J) = ASI
99 CONTINUE
RETURN
SUBROUTINE ELHLLIB

SUBPROGRAM ELHLLIB CALCULATES THE SHAPE FUNCTIONS AND THE PARTIAL DERIVATIVES OF THE SHAPE FUNCTIONS WITH THE LOCAL COORDINATES 'XI', 'ETA' AND 'SI'.

<table>
<thead>
<tr>
<th>FUNCTION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>H(I)</td>
<td>SHAPE FUNCTION OF THE ELEMENT</td>
</tr>
<tr>
<td>HXI(I)</td>
<td>PARTIAL DERIVATIVE OF 'H' WRT 'XI'</td>
</tr>
<tr>
<td>ETA(I)</td>
<td>PARTIAL DERIVATIVE OF 'H' WRT 'ETA'</td>
</tr>
<tr>
<td>SSI(I)</td>
<td>PARTIAL DERIVATIVE OF 'H' WRT 'SI'</td>
</tr>
</tbody>
</table>

**Switches:**
- RENUMB=100:10, FORMAT=900:10

**Functions called from this routine:**
- ELMEXT

**REAL**
- XI, ETA, SI, H, HXI, ETAI, SSI, XI0, ETA0, SSI0

**COMMON**
- ELLLIB/XII(20), ETAI(20), SSI(20)
- DIMENSION H(20), HXI(20), ETAI(20), SSI(20)

**ENTRY ISOP2D(XI, ETA, HXI, ETAI, ITYPE, IERROR)**

--- DRIVATIVE OF SHAPE FUNCTIONS FOR 2D ISOPARAMETRIC QUADRILATERAL ELEMENTS

- HXI(1) = -0.25*(1.-ETA)
- HXI(2) = 0.25*(1.-ETA)
- HXI(3) = 0.25*(1.+ETA)
- HXI(4) = -0.25*(1.+ETA)
- ETAI(1) = -0.25*(1.-XI)
- ETAI(2) = -0.25*(1.+XI)
- ETAI(3) = 0.25*(1.+XI)
- ETAI(4) = 0.25*(1.-XI)

**IF (ITYPE .EQ. 219) THEN**

--- DRIVATIVE OF SHAPE FUNCTIONS FOR 9 NODES SERENDIPITY ISOPARAMETRIC ELEMENT

- CST = 9. DO/25. DO
- C13 = 1. DO/3. DO
- C23 = 2. DO/3. DO
- HXI(5) = CST*(1.-ETA)*((9.*XI**2-2.*XI-3.)
- HXI(6) = CST*(1.-ETA)*((-9.*XI**2)-2.*XI+3.)
- HXI(7) = 0.5*(1.-ETA)**2
- HXI(8) = -XI*(1.+ETA)
\[ H(0) = -0.5 \times (1. - \eta^2) \]
\[ H(1) = H(1) - 0.5 \times H(3) - C23 \times H(5) - C13 \times H(6) \]
\[ H(2) = H(2) - 0.5 \times (H(7) + H(8)) \]
\[ H(3) = H(3) - 0.5 \times (H(7) + H(8)) \]
\[ H(4) = H(4) - 0.5 \times (H(7) + H(8)) \]
\[ H(5) = H(5) - 0.5 \times (H(7) + H(8)) \]
\[ H(6) = H(6) - 0.5 \times (H(7) + H(8)) \]
\[ H(7) = H(7) - 0.5 \times (H(7) + H(8)) \]
\[ H(8) = H(8) - 0.5 \times (H(7) + H(8)) \]

**IF** (ITYPE .EQ. 204) **GO TO** 100

**C** ADDITIONAL TERMS FOR THE FIVE NODE ISOPARAMETRIC QUADRILATERAL ELEMENT.

\[ H(6) = H(6) - 0.5 \times (1. - \xi^2) \]
\[ H(7) = H(7) - 0.5 \times (1. - \xi^2) \]
\[ H(8) = H(8) - 0.5 \times (1. - \xi^2) \]
\[ H(9) = H(9) - 0.5 \times (1. - \xi^2) \]
\[ H(10) = H(10) - 0.5 \times (1. - \xi^2) \]

**IF** (ITYPE .EQ. 205) **GO TO** 100

**C** ADDITIONAL TERMS FOR THE EIGHT NODE ISOPARAMETRIC QUADRILATERAL ELEMENT.

\[ H(6) = H(6) - 0.5 \times (1. - \xi^2) \]
\[ H(7) = H(7) - 0.5 \times (1. - \xi^2) \]
\[ H(8) = H(8) - 0.5 \times (1. - \xi^2) \]
\[ H(9) = H(9) - 0.5 \times (1. - \xi^2) \]
\[ H(10) = H(10) - 0.5 \times (1. - \xi^2) \]
\[ H(11) = H(11) - 0.5 \times (1. - \xi^2) \]
\[ H(12) = H(12) - 0.5 \times (1. - \xi^2) \]
\[ H(13) = H(13) - 0.5 \times (1. - \xi^2) \]
\[ H(14) = H(14) - 0.5 \times (1. - \xi^2) \]
\[ H(15) = H(15) - 0.5 \times (1. - \xi^2) \]
\[ H(16) = H(16) - 0.5 \times (1. - \xi^2) \]
\[ H(17) = H(17) - 0.5 \times (1. - \xi^2) \]
\[ H(18) = H(18) - 0.5 \times (1. - \xi^2) \]
\[ H(19) = H(19) - 0.5 \times (1. - \xi^2) \]
\[ H(20) = H(20) - 0.5 \times (1. - \xi^2) \]

**IF** (ITYPE .EQ. 208) **GO TO** 100

**C** ADDITIONAL TERMS FOR THE EIGHT NODE LAGRANGIAN ISOPARAMETRIC QUADRILATERAL ELEMENT.

\[ H(6) = H(6) - 0.5 \times (1. - \xi^2) \]
\[ H(7) = H(7) - 0.5 \times (1. - \xi^2) \]
\[ H(8) = H(8) - 0.5 \times (1. - \xi^2) \]
\[ H(9) = H(9) - 0.5 \times (1. - \xi^2) \]
\[ H(10) = H(10) - 0.5 \times (1. - \xi^2) \]
\[ H(11) = H(11) - 0.5 \times (1. - \xi^2) \]
\[ H(12) = H(12) - 0.5 \times (1. - \xi^2) \]
\[ H(13) = H(13) - 0.5 \times (1. - \xi^2) \]
\[ H(14) = H(14) - 0.5 \times (1. - \xi^2) \]
\[ H(15) = H(15) - 0.5 \times (1. - \xi^2) \]
\[ H(16) = H(16) - 0.5 \times (1. - \xi^2) \]
\[ H(17) = H(17) - 0.5 \times (1. - \xi^2) \]
\[ H(18) = H(18) - 0.5 \times (1. - \xi^2) \]
\[ H(19) = H(19) - 0.5 \times (1. - \xi^2) \]
\[ H(20) = H(20) - 0.5 \times (1. - \xi^2) \]

**ELSE**

**IF** (ITYPE .EQ. 204) **GO TO** 100

**C** ADDITIONAL TERMS FOR THE FIVE NODE ISOPARAMETRIC QUADRILATERAL ELEMENT.

\[ H(6) = H(6) - 0.5 \times (1. - \xi^2) \]
\[ H(7) = H(7) - 0.5 \times (1. - \xi^2) \]
\[ H(8) = H(8) - 0.5 \times (1. - \xi^2) \]
\[ H(9) = H(9) - 0.5 \times (1. - \xi^2) \]

**IF** (ITYPE .EQ. 205) **GO TO** 100

**C** ADDITIONAL TERMS FOR THE EIGHT NODE ISOPARAMETRIC QUADRILATERAL ELEMENT.

\[ H(6) = H(6) - 0.5 \times (1. - \xi^2) \]
\[ H(7) = H(7) - 0.5 \times (1. - \xi^2) \]
\[ H(8) = H(8) - 0.5 \times (1. - \xi^2) \]
\[ H(9) = H(9) - 0.5 \times (1. - \xi^2) \]

**IF** (ITYPE .EQ. 208) **GO TO** 100

**C** ADDITIONAL TERMS FOR THE EIGHT NODE LAGRANGIAN ISOPARAMETRIC QUADRILATERAL ELEMENT.

\[ H(6) = H(6) - 0.5 \times (1. - \xi^2) \]
\[ H(7) = H(7) - 0.5 \times (1. - \xi^2) \]
\[ H(8) = H(8) - 0.5 \times (1. - \xi^2) \]
\[ H(9) = H(9) - 0.5 \times (1. - \xi^2) \]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\( \beta(2) = \beta(2) + \beta(9)/4. \)
\( \beta(3) = \beta(3) + \beta(9)/4. \)
\( \beta(4) = \beta(4) + \beta(9)/4. \)
\( \beta(5) = \beta(5) - \beta(9)/2. \)
\( \beta(6) = \beta(6) - \beta(9)/2. \)
\( \beta(7) = \beta(7) - \beta(9)/2. \)
\( \beta(8) = \beta(8) - \beta(9)/2. \)

ENDIF

100 CONTINUE

RETURN

ENTRY N2D (XI, ETA, N, ITYPE, IERROR)

C ---- SHAPE FUNCTIONS FOR 2D QUADRILATERAL ISOPARAMETRIC ELEMENT.

C

N(1) = 0.25*(1.-XI)*(1.-ETA)
N(2) = 0.25*(1.-XI)*(1.-ETA)
N(3) = 0.25*(1.+XI)*(1.+ETA)
N(4) = 0.25*(1.+XI)*(1.+ETA)

C

IF (ITYPE .EQ. 219) THEN

C ---- SHAPE FUNCTIONS FOR 9 NODES SEREDIPITY ISOPARAMETRIC ELEMENT.

C

CST = 9.DO/32.DO
C13 = 1.DO/3.DO
C23 = 2.DO/3.DO
N(5) = CST*(1.-ETA)*(1.-XI)**2*(1.-3.*XI)
N(6) = CST*(1.-ETA)*(1.-XI)**2*(1.+3.*XI)
N(7) = 0.5*(1.-XI)**2*(1.-ETA)**2
N(8) = 0.5*(1.-XI)**2*(1.+ETA)
N(9) = 0.5*(1.-XI)**2*(1.-ETA)
N(10) = B(1) - 0.5*H(9) - C23*B(B) - C13*B(B)
N(11) = B(2) - 0.5*H(7) - C23*B(B) - C13*B(B)
N(12) = B(3) - 0.5*H(6) - C23*B(B) - C13*B(B)
N(13) = B(4) - 0.5*(H(8)+H(9))
ELSE

IF (ITYPE .EQ. 204) GO TO 110

C ---- ADDITIONAL TERMS FOR THE FIVE NODED SEREDIPITY ELEMENT

C

N(5) = 0.5*(1.-XI)**2*(1.-ETA)
N(1) = H(1) - 0.5*H(5)
N(2) = H(2) - 0.5*H(5)

C

IF (ITYPE .EQ. 205) GO TO 110

C ---- ADDITIONAL TERMS FOR THE EIGHT NODED SEREDIPITY ELEMENT

C

N(6) = 0.5*(1.-XI)**2*(1.-ETA)
N(7) = 0.5*(1.-XI)**2*(1.+ETA)
N(8) = 0.5*(1.-XI)**2*(1.+ETA)
N(1) = H(1) - 0.5*H(8)
N(2) = H(2) - 0.5*H(8)
N(3) = H(3) - 0.5*(H(7)+H(6))
N(4) = H(4) - 0.5*(H(7)+H(8))

C

IF (ITYPE .EQ. 208) GO TO 110
C ---- ADDITIONAL TERMS FOR THE NINE NODED LAGRANGIAN ELEMENT

C

H(9) = (1.-ETA**2)*(1.-XI)**2
H(1) = H(9) + H(9)/4.
H(2) = H(2) + H(9)/4.
H(3) = H(3) + H(9)/4.
H(5) = H(5) - H(9)/2.
H(6) = H(6) - H(9)/2.
H(7) = H(7) - H(9)/2.
H(8) = H(8) - H(9)/2.

ENDIF

110 CONTINUE
RETURN

C ---- SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR THE 3D ISOP. EL.

C

ENTRY ISO3PD (XI, ETA, SI, N, XI, ETA, SI, ITYPE, ERROR)

C

WRITE(*,*) 'ISO3PD ENTERED'
IF (ITYPE .EQ. 308) THEN

DO 120 K = 1, 8
CALL ELMEXT (XI, ETA, SI, K, XI, ETA, SI)
B(K) = 0.125*(1.+XI)*(1.+ETA)*(1.+SII)
BXI(K) = 0.125*(1.+ETA)*(1.+SI)**2*XII(K)
BETA(K) = 0.125*(1.+XI)*(1.+SI)*ETAI(K)
BSI(K) = 0.125*(1.+XI)*(1.+ETA)**2*SI(K)

120 CONTINUE
ELSE IF (ITYPE .EQ. 320) THEN

C HEXAHERO SOLID ELEMENT

C SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR NODES 1-8.

C

DO 130 K = 1, 8
CALL ELMEXT (XI, ETA, SI, K, XI, ETA, SI)
H(K) = 0.125*(1.+XI)*(1.+ETA)*(1.-SI)**2
BXI(K) = 0.125*(1.+ETA)*(1.-SI)**2*XII(K)
BETA(K) = 0.125*(1.+XI)*(1.-SI)**2*ETAI(K)
BSI(K) = -0.125*(1.+ETA)*(1.-SI)**2*SI(K)

130 CONTINUE

C K1 = 9
K2 = 10

C SHAPF FUNCTION AND THEIR DERIVATIVES FOR NODES 13-16.

C

DO 140 K = 13, 16
CALL ELMEXT (XI, ETA, SI, K, XI, ETA, SI)
H(K) = 0.125*(1.+XI)*(1.-SI)**2
BXI(K) = 0.125*(1.+XI)*(1.-SI)**2*XII(K)
BETA(K) = 0.125*(1.+XI)*(1.-SI)**2*ETAI(K)
BSI(K) = -0.125*(1.+ETA)*(1.+XI)*SI(K)

C SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR NODES 9,11,17,19.

C

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CALL ELMEXT (XI, ETA, SI, K1, XIO, ETAO, SIO)
     H(K1) = .25*(1.-XI**2)*(1.+SI0)*XI + ETA0)*SI0)
     H XI(K1) = -0.6*(1.+ETAO)*(1.+SIO)*XI
     H ETA(K1) = .25*(1.-XI**2)*(1.+SIO)*ETA1(K1)
     H SI(K1) = .25*(1.+ETAI)*XI**2)*SI1(K1)

SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR NODES 10,12,18,20.

CALL ELMEXT (XI, ETA, SI, K2, XIO, ETAO, SIO)
     H(K2) = .25*(1.+XI0)**2)*(1.-ETA**2)*(1.+SI0)
     H XI(K2) = .25*(1.-ETA**2)*(1.+SIO)*XI
     H ETA(K2) = -0.5*(1.+XI0)*(1.+SI0)*ETA
     H SI(K2) = .25*(1.-ETA**2)*(1.+XI0)*SI1(K2)

IF (K1 .EQ. 11) THEN
     K1 = 17
     K2 = 18
ELSE
     K1 = K1 + 2
     K2 = K2 + 2
ENDIF

140 CONTINUE
C
C .. SWITCHES: REUMB=100;10,FORMAT=900;10
C .. SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
REAL*8 XI, ETA, SI, XIO, ETAO, SIO
COMMON/ELLIB1/XII(20), ETAI(20), SII(20)
C
C XI = XI**2(K)
ETAO = ETAI*ETAI(K)
SI0 = SI**2(K)

RETURN
END
C
C INCLUDE (PROCESS)
C SUBROUTINE JACOBI
C
C INCLUDE (PROCESS)
C SUBROUTINE JACOBI
C
C INCLUDE (PROCESS)
C SUBROUTINE JACOBI
C
C INCLUDE (PROCESS)
C SUBROUTINE JACOBI
This program calculates the Jacobian of the transformation between the local coordinates \( X \) and \( \eta \) and the global coordinates \( x \) and \( y \) for integration point \( 'IHTGPN' \) of element number \( 'HREL' \).

\[
\begin{align*}
\text{HREL} &= \text{ELEMENT NUMBER} \\
\text{DETJAC} &= \text{DETERMINANT OF THE JACOBIAN} \\
\text{BXXI(I)} &= \text{PARTIAL DERIVATIVE OF } X(I) \text{ WITH RESPECT TO } X \\
\text{BXY(I)} &= \text{PARTIAL DERIVATIVE OF } X(I) \text{ WITH RESPECT TO } Y \\
\text{BXZ(I)} &= \text{PARTIAL DERIVATIVE OF } X(I) \text{ WITH RESPECT TO } Z \\
\text{ERROR} &= \text{ERROR CODE FOR EXCESSIVE ELEMENT DISTORTION} \\
\text{BXX(I)} &= \text{NODE COORDINATE } X, \text{ IX=GLOBAL NODE NUMBER} \\
\text{BXY(I)} &= \text{NODE COORDINATE } Y, \text{ IX=GLOBAL NODE NUMBER} \\
\text{BZX(I)} &= \text{NODE COORDINATE } Z, \text{ IX=GLOBAL NODE NUMBER} \\
\end{align*}
\]

\[
\begin{align*}
\text{XXI} &= \text{PARTIAL DERIVATIVE OF } X \text{ WITH RESPECT TO } X \\
\text{XXY} &= \text{PARTIAL DERIVATIVE OF } X \text{ WITH RESPECT TO } Y \\
\text{XXZ} &= \text{PARTIAL DERIVATIVE OF } X \text{ WITH RESPECT TO } Z \\
\text{XYY} &= \text{PARTIAL DERIVATIVE OF } Y \text{ WITH RESPECT TO } X \\
\text{XYZ} &= \text{PARTIAL DERIVATIVE OF } Y \text{ WITH RESPECT TO } Y \\
\text{XZZ} &= \text{PARTIAL DERIVATIVE OF } Y \text{ WITH RESPECT TO } Z \\
\end{align*}
\]

IMPLICIT REAL*8 (A-H, O-Z)

SWITCHES: 
- REHB=100:10, FORMAT=900:10
- SWITCHES:

SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

ERRORS

C

REAL*8 E,BXXI,BXYI,BXZI,EX,EX,IZ
REAL*4 XYZ
CHARACTER COMM*6, BUFFER*80, BUFF*80, CELEM*7
COMM/RSHAPE/8(20,27),BXXI(20,27),BXYI(20,27),BXZI(20,27),SI(27)
COMM/SHELDR/VECT(3,3,9)
COMM/INPUT2/6D0(20,6000)
COMM/INPUT3/XYZ(3,10000)
COMM/JACOBI/WX(20),WY(20),WX(20),Y(20),Y(20),X(20),SI,SYI,SIZ
COMM/CMP2/CMP, BUFFER, BUFF
DATA ZERO/0.000, CELEM/ 'ELEMENT'/
DIMENSION V3( 3 )
SAVE ZERO, CELEM
C
ENTRY JACB2D(IHTGPN,HREL,BREL,ERROR,DETJAC)
XXI = ZERO
XETA = ZERO
YXI = ZERO
YETA = ZERO
C
C*VDIR: ASSUME COUNT(S)
C
DO 100 K1 = 1, NREL
    NODE = NCP(K1,NREL)
    X = XVZ(NODE)
    Y = YVZ(NODE)
    XXI = XXI + XXI(K1,NTOPS)*X
    XETA = XETA + XETA(K1,NTOPS)*Y
    YXI = YXI + YXI(K1,NTOPS)*X
    YETA = YETA + YETA(K1,NTOPS)*Y
100 CONTINUE
C
DETJAC = XXI*YETA - YXI*XETA
DETJ = DETJAC
C
C ----- CHECK THE INTEGRITY OF THE TRANSFORMATION FROM ISOP. COORD.
C TO THE GLOBAL COORDINATES
C
IF (DETJAC .EQ. ZERO) THEN
    WRITE (BUFFER, *) CELEM, HREL
    CALL ERRORS (16, 1, 'JACB2D')
C
C ----- DETERMINANT OF JACOBIAN AS EVALUATED ABOVE WILL BE NEGATIVE IF
C NODE NUMBERING IS DONE IN THE CLOCKWISE DIRECTION. TO HAVE
C CONSISTENT TRANSFORMATIONS WITH RESPECT TO A 3-D SPACE THE
C DETERMINANT OF JACOBIAN IS MADE POSITIVE, HOWEVER X1 AND Y1 ARE
C EVALUATED USING THE NEGATIVE DETERMINANT SAVED IN DETJ VARIABLE.
C
ELSE IF (DETJAC .LT. ZERO) THEN
    DETJAC = -DETJAC
ENDIF
C
C*VDIR: ASSUME COUNT(S)
C
DO 110 K = 1, NREL
    DX(K) = (YETA*XXI(K,NTOPS)-YXI*XETA(K,NTOPS))/DETJ
    DY(K) = ((-XETA*XXI(K,NTOPS)+XXI*XETA(K,NTOPS))/DETJ
110 CONTINUE
RETURN
C
ENTRY JACB3D (INTOPS, NREL, NREL, NERROR, DETJAC)
C
WRITE(*,*) 'JACB3D CALLED'
XXI = ZERO
XETA = ZERO
YXI = ZERO
YETA = ZERO
YSI = ZERO
ZXI = ZERO
ZETA = ZERO
ZSI = ZERO
C
C*VDIR: ASSUME COUNT(20)
C
DO 120 K1 = 1, NREL
    NODE = NCP(K1,NREL)
X = XYZ(1, NODE)
Y = XYZ(2, NODE)
Z = XYZ(3, NODE)
XII = X + XI*XI(k, IETOPB)*X
YII = Y + YI*XI(k, IETOPB)*Y
ZII = Z + ZI*XI(k, IETOPB)*Z
XETA = XETA + HET(r, IDTGPII)*X
YETA = YETA + HET(r, IDTGPII)*Y
ZETA = ZETA + HET(r, IDTGPII)*Z
XSI = XSI + HSI(r, IDTGPII)*X
YSI = YSI + HSI(r, IDTGPII)*Y
ZSI = ZSI + HSI(r, IDTGPII)*Z
120 CONTINUE
C
DETJAC = X*(YETA*ZSI-ZETA*YSI) - YXI*(XETA*ZSI-ZETA*XSI) + ZXI*(XETA*YSI-YETA*XSI)
C
C ----- CHECK THE INTEGRITY OF THE TRANSFORMATION FROM ISOP. COORD.
C TO THE GLOBAL COORDINATES
C
IF (DETJAC .EQ. ZERO) THEN
WRITE (BUFFER, *) CELEH, NREL
CALL ERRORS (16, 1, 'JACB3D')
ENDIF
C
C*VDIR: ASSUME C0UNTC2O)
C
DO 130 K = 1, NNEL
HX(K) = ((YETA*ZSI-ZETA*YSI)*NXI(K, INTGPN>-(YXI*ZSI-ZXI*YSI)*NETA (K, INTGPH) + (YXI*ZETA-ZXI*XETA) *NSI (K, INTGPN) )/DETJAC
NY(K) = ((-(XETA*ZSI-ZETA*XSI)»NXI(K,INTGPN))+(XXI*ZSI-ZXI*XSI ) )*HETA (K , INTGPH)- (XXI*ZETA-ZXI*XETA) •N SK K , INTGPN) ) /DETJAC
BZ(K) = ((XETA»YSI-YETA»XSI)»HXI(K,IHTGPD)-(XXI*YSI-YXI»XSI)* NETACK,INTGPn) + (XXI*YETA-YXI»XETA)*NSI(K,INTGPH) )/DETJAC
130 CONTINUE
RETURN
C
C
ENTRY JACSHL (INTGPN, NREL, NNEL, THICK, DETJAC, VS, ISET, SIP)
C
WRITE(*,*) 'INTO JACSHL', 'INTGPN', 'INTGPN', 'NNEL', NNEL, 'NREL', NREL, 'THICK', THICK
C
$ XLI = ZERO
XETA = ZERO
XSI = ZERO
YII = ZERO
YETA = ZERO
YSI = ZERO
ZII = ZERO
ZETA = ZERO
ZSI = ZERO
C
XII = ZERO
XETA = ZERO
YII = ZERO
YETA = ZERO
ZII = ZERO
ZETA = ZERO
ZSI = ZERO
C
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
VALUE = ZERO

C
C*VDIR: ASSUME CQUST(8)
C
C ------ FIND THE DERIVATIVES OF X,Y AND Z WITH RESPECT TO THE
C ISOPARAMETRIC COORDINATES. ALSO EVALUATE THE NORMAL TO THE
C MID PLANE OF SHELL BY INTERPOLATING THE NODALNormals THROUGH
C THE ELEMENT SHAPE FUNCTIONS. ISET=0
C
C IF ISET=1 FIND NORMAL AT THE TOP OR BOTTOM SURFACE.
IF (ISET .EQ. 0) VALUE = 0.0
IF (ISET .EQ. 1) VALUE = 1.0
CONST1 = THICKx0.600
CONST2 = SIPxCONST1
C
WHITE(*,*) 'SIP', SIP, 'CQUST2', 'CQUST1', 'CQUST1'
DO 140 XI = 1, N_NEL

NODE = NODE(K1,NREL)
X = XYZ(1,NODE)
Y = XYZ(2,NODE)
Z = XYZ(3,NODE)

XXI = XXI + NXI(K1,INTGPN) * (X+VALUExCONST2xVECT(1,3,X1))
YXI = YXI + HXI(K1,INTGPN) * (Y+VALUExCONST2xVECT(2,3,XI))
ZXI = ZXI + HXI(K1,INTGPN) * (Z+VALUExCONST2xVECT(3,3,XI))
XETA = XETA + NETA(XI,INTGPN) * (X+VALUExCONST2xVECT(1,3,X1))
YETA = YETA + NETA(XI,INTGPN) * (Y+VALUExCONST2xVECT(2,3,XI))
ZETA = ZETA + NETA(XI,INTGPN) * (Z+VALUExCONST2xVECT(3,3,X1))

XSI = XSI + N(XI,INTGPN)xVECT(1,3,X1)
YSI = YSI + N(XI,INTGPN)xVECT(2,3,XI)
ZSI = ZSI + N(XI,INTGPN)xVECT(3,3,XI)
C
C ------ EVALUATE TWO VECTORS TANGENT TO THE MID SURFACE (SI=0)
C THESE VECTORS ARE V1={(XXI, YXI, ZXI), V2={(XETA, YETA, ZETA)}
C
XXI = XXI + XI(K1,INTGPN) * (X+VALUExCONST2xVECT(1,3,X1))
YXI = YXI + XI(K1,INTGPN) * (Y+VALUExCONST2xVECT(2,3,XI))
ZXI = ZXI + XI(K1,INTGPN) * (Z+VALUExCONST2xVECT(3,3,X1))
XETA = XETA + XI(K1,INTGPN) * (X+VALUExCONST2xVECT(1,3,X1))
YETA = YETA + XI(K1,INTGPN) * (Y+VALUExCONST2xVECT(2,3,XI))
ZETA = ZETA + XI(K1,INTGPN) * (Z+VALUExCONST2xVECT(3,3,X1))

140 CONTINUE
C
C ------ EVALUATE THE CROSS PRODUCT OF V1 AND V2 TO FIND THE NORMAL
C VECTOR V3
C
V3(1) = YXIxZETA1 - YETA1xXXI
V3(2) = ZXIxZETA1 - XXIxYETA1
V3(3) = XXIxYETA1 - YXIxXETA1
IF (ISET .EQ. 0) THEN

V3(1) = SQRT(V3(1)**2+V3(2)**2+V3(3)**2)
V3(1) = V3(1)/V3RM
V3(2) = V3(2)/V3RM
V3(3) = V3(3)/V3RM
ENDIF
C
DETJAC = XXI*(YETAxZSI-ZETAxYSI) - YXI*(ZETAxZSI-ZETAxXSI) + ZXI*(
XETAxYSI-YETAxXI)
C
C ------ CHECK THE INTEGRITY OF THE TRANSFORMATION FROM ISOP. COORD.
C TO THE GLOBAL COORDINATES
C
IF (DETJAC .LE. ZERO) THEN

WRITE (BUFFER, *) CELEM, NREL
CALL ERRORS (16, 1, 'JACSHL')
C
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C*VDIR: ASSUME COUNT(20)
C
DO 160 K = 1, NEL
   HX(K) = ((YETA*ZSI-ZEIA*YSI)*BXI(K,IVGPH)-(YXI*ZSI-ZXI*YSI)*
             1     )/DETJAC
   HY(K) = (((-IETA»ZSI-ZETI*ISI)*HXI(K,ITGPH)) + (XXI»ZSI-ZII»XSI
             1     )  »BETA (K.BTGPH))/DETJAC
   BZ(K) = ((XETA»YSI-YETA«XSI)»BXI(K,IBTGPB)-(XXI*YSI-YXI*XSI)*
              1     )/DETJAC
   RETURN
1 CONTINUE

IBO COBTIBUE

SIX = (YXI*ZETA-ZXI*YETA)/DETJAC
SIY = -(XXI»ZETA-ZXI«XETA)/DETJAC
SIZ = (IXI»YETA-YXI*XETA)/DETJAC
RETURN
END

C* ============ S H B 0 R M =============
C* INCLUDE (PROCESS)
SUBROUTINE SHORM(ELBUN, DO, BEEL, ITYPE, V1, V2, V3)
C
I
C SHORM EVALUATES THE X, Y, AND Z DIRECTIONS
C AT THE ELEMENT NODES FOR GENERAL SHELL ELEMENTS
C
I
C ARGUMENT LIST
C
I
C ELBUN = ELEMENT NUMBER
C DO = ELEMENT NODE NUMBER
C BEEL = NUMBER OF NODES IN THE ELEMENT
C ITYPE = ELEMENT TYPE
C V1 = VECTOR CONTAINING THE DIRECTION COSINES OF THE X-DIRECTION
C V2 = VECTOR CONTAINING THE DIRECTION COSINES OF THE Y-DIRECTION
C V3 = VECTOR CONTAINING THE DIRECTION COSINES OF THE Z-DIRECTION
C
I
C IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: REBUN=100:10, FORMAT=900:10
C...SWITCHES:
C*============ SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE ============
C
C ISOP2D DIRVEC
C
C REAL*4 XYZ,PXI,PETA
INTEGER ELBUN
COMMON/INPUT2/BEP(20,6000)
COMMON/INPUT3/XYZ(3,10000)
COMMON/ELLIB2/PXI9),PETA(9)
DIMENSION V1(3 ),V2(3 ),V3(3 ),PIXI(20),PETA(20)
C
IT = ITYPE - 100

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
AXI = PXI(ND)
AETA = PETA(ND)
CALL ISOP2D (AXI, AETA, FXI, FETA, IT, IERROR)

XII = 0.0
XETA = 0.0
XSI = 0.0
YXI = 0.0
YETA = 0.0
YSI = 0.0
ZXI = 0.0
ZETA = 0.0
ZSI = 0.0

C
C*VDIR: ASSUM E COUHT(S)
C
C EVALUATE TWO VECTORS TANGENT TO THE MID SURFACE (SI=0)
C THESE VECTORS ARE V1={XXI, YXI, ZXI}, V2={XETA, YETA, ZETA}
C
DO 100 K1 = 1, NNEL
   NODE = NCP(K1, LNUM)
   X = XYZC(NODE)
   Y = XYZC(NODE)
   Z = XYZC(NODE)
   X XI = XXI + FXI(K1) * X
   Y XI = YXI + FXI(K1) * Y
   Z XI = ZXI + FXI(K1) * Z
   X ETA = XETA + FETA(K1) * X
   Y ETA = YETA + FETA(K1) * Y
   Z ETA = ZETA + FETA(K1) * Z
100 CONTINUE

C ***** EVALUATE THE CROSS PRODUCT OF V1 AND V2 TO FIND THE NORMAL C VECTOR V3
C
V3(1) = Y XI * Z ETA - Y ETA * ZXI
V3(2) = ZXI * X ETA - XXI * Z ETA
V3(3) = XXI * Y ETA - Y XI * X ETA
V3ORM = DSQRT(V3(1)**2 + V3(2)**2 + V3(3)**2)
V3(1) = V3(1) / V3ORM
V3(2) = V3(2) / V3ORM
V3(3) = V3(3) / V3ORM

C
C ***** EVALUATE THE COMPLETE ORTHONORMAL COORDINATE SYSTEM.
C V1 AND V2 EVALUATED NEXT WILL BE DIFFERENT FROM WHAT WAS C PREVIOUSLY DESCRIBED.
C
CALL DIRVEC (V1, V2, V3)

C RETURN
END

C******************************************************************************
C INCLUDE (PROCESS)
SUBROUTINE ELINFO(ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
C******************************************************************************
C PROGRAM 'ELINFO' EXTRACTS ELEMENT INFORMATION FROM THE ARRAY
The text contains a list of argument definitions for a subroutine or function, followed by a block of Fortran code. The arguments include:

- `ELNUM` = Element number passed by the calling routine
- `ITYPE` = Element type passed to the calling routine
- `NBEL` = Number of nodes in the element passed to the calling routine
- `MATNUM` = Material I.D. number for the element passed to the calling routine
- `ISTART` = Starting position of the line connectivity data in arrays 'IS' and 'IE'.
- `LINES` = Number of lines connecting the nodes within the element.

The code snippet includes declarations for arrays and variables, and a function equivalence statement. It also mentions that the subroutine or function is called 'ELINFO CALLED'.
384

C  8-14  255  #. OF LAYERS
C  15-16  3  INTERNAL LAYER FLAG.
C  17-22  63  NUMBER OF LINES CONNECTING NODES
C  23-29  128  STARTING POSITION OF LINE
C  308  8  CONNECTIVITY IN ARRAYS ISTART & IEND
C  
C  ---- TABLE OF INTERNAL ELEMENT TYPE NUMBERS
C  TYPE NO.  NO. OF NODES  DESCRIPTION
C  204  4  LINEAR ISOPARAMETRIC QUADRILATERAL
C  206  8  QUADRATIC ISOPARAMETRIC QUADRILATERAL
C  209  9  LAGRANGIAN QUADRATIC ISOP. QUAD.
C  219  9  QUADRATIC TO QUIC B QUAD.
C  308  8  LINEAR ISOPARAMETRIC BRICK
C  319  20  QUADRATIC ISOPARAMETRIC BRICK
C
I = INFOEL(1,ELNUM)
I1 = INFOEL(2,ELNUM)
LINES = IAND(I1,65535)/131072
ISTART = I1/65536
IFLAG = IAND(I,1331664)/262144
ITYPE = IAND(I,5453984)/512
NNEL = IAND(I,508)/8
RETURN

ENTRY ELINTM(ELNUM,IDENT,INTCOD,NIPXI,NIPETA,NIPSI,MATNUM,THICK)
I = INFOEL(1,ELNUM)
I1 = INFOEL(2,ELNUM)
NNEL = IAND(I,508)/8
MATNUM = IAND(I,7)
NIPXI = I/134217728
NIPETA = IAND(I,177440512)/16777216
NIPSI = IAND(I,16680064)/2097152
INTCOD = IAND(I,256)
IDENT = INTCOD + NIPXI + NIPETA*10 + NIPSI*100
NLAYRS = INT(ELLYFO(320,ELNUM))
IF (IFLAG.EQ.1 .OR. IFLAG.EQ.2 .OR. IFLAG.EQ.4) THEN
  ITH = INFOEL(3,ELNUM)
  THICK(1) = TH
  ITH = INFOEL(4,ELNUM)
  THICK(2) = TH
  ITH = INFOEL(5,ELNUM)
  THICK(3) = TH
  ITH = INFOEL(6,ELNUM)
  THICK(4) = TH
  IF (NNEL .GT. 4) THEN
    THICK(5) = (THICK(1)+THICK(2))/0.8
    THICK(6) = (THICK(2)+THICK(3))/0.8
    THICK(7) = (THICK(3)+THICK(4))/0.8
    THICK(8) = (THICK(4)+THICK(5))/0.8
  ENDIF
  0.25
ENDIF
END

C=======================================
C INCLUDE(PROCESS)
SUBROUTINE LYINFO(ILAYER,LTHICK,25,MATRL,DCS,NNEL,
$ IMPLICIT REAL=8(1-E,0-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*****************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C*****************************************************************************
C
INTEGER ELHUM
REAL=LTHICK
COMMON/LAYER/ELLYFO(320,5000)
DIMENSION LTHICK(9),ZS(9),DCS(3,3)

INDEX = 21*(ILAYER-1) + 1
MATRL = ELLYFO(INDEX,ELHUM)
LTHICK(1) = ELLYFO(INDEX+1,ELHUM)
LTHICK(2) = ELLYFO(INDEX+2,ELHUM)
LTHICK(3) = ELLYFO(INDEX+3,ELHUM)
LTHICK(4) = ELLYFO(INDEX+4,ELHUM)
IF (HELM .GT. 4) THEN
LTHICK(5) = (LTHICK(1)+LTHICK(2))*0.50
LTHICK(6) = (LTHICK(2)+LTHICK(3))*0.50
LTHICK(7) = (LTHICK(3)+LTHICK(4))*0.50
LTHICK(8) = (LTHICK(4)+LTHICK(1))*0.50
LTHICK(9) = (LTHICK(1)+LTHICK(2))*0.25 + (LTHICK(3)+LTHICK(4))
ENDIF

ZS(1) = ELLYFO(INDEX+5,ELHUM)
ZS(2) = ELLYFO(INDEX+6,ELHUM)
ZS(3) = ELLYFO(INDEX+7,ELHUM)
ZS(4) = ELLYFO(INDEX+8,ELHUM)
IF (HELM .GT. 4) THEN
ZS(5) = (ZS(1)+ZS(2))*0.60
ZS(6) = (ZS(2)+ZS(3))*0.60
ZS(7) = (ZS(3)+ZS(4))*0.60
ZS(8) = (ZS(4)+ZS(1))*0.60
ZS(9) = (ZS(1)+ZS(2))*0.25 + (ZS(3)+ZS(4))*0.25
ENDIF

DCS(1,1) = ELLYFO(INDEX+9,ELHUM)
DCS(1,2) = ELLYFO(INDEX+10,ELHUM)
DCS(1,3) = ELLYFO(INDEX+11,ELHUM)
DCS(2,1) = ELLYFO(INDEX+12,ELHUM)
DCS(2,2) = ELLYFO(INDEX+13,ELHUM)
DCS(2,3) = ELLYFO(INDEX+14,ELHUM)
DCS(3,1) = ELLYFO(INDEX+15,ELHUM)
DCS(3,2) = ELLYFO(INDEX+16,ELHUM)
DCS(3,3) = ELLYFO(INDEX+17,ELHUM)

HIPXI = INT(ELLYFO(INDEX+18,ELHUM))
HIPETA = INT(ELLYFO(INDEX+19,ELHUM))
HIPSI = INT(ELLYFO(INDEX+20,ELHUM))

RETURN
END
SUBROUTINE FILES
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***************************************************************
C C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C FILEISF
C***************************************************************

CHARACTER*40 SLASH
CHARACTER*40 SCI,SC2,SC3,SC4,SC6,SC7,OUT,STAT,PLT,IHT,CRR
CHARACTER*40 IH1,IH2,IH3,STY,ST1,ST2,SPG,SG1,SG2,CK1,AFK
CHARACTER*40 IH4,IH5,IH6,IH7,IH8,IH9,IHA,IHB,IHC,IHD,IHE,IHF
CHARACTER*40 SCFP,OFP,OFH,VOLSER,VOL
LOGICAL*4 THERE
COMMON/FILE1/KEEP,DEL,SCFP,OFP,OFH,VOLSER
COMMON/INPUT//FILE4,IOINTH,IFPLOT
DATA SLASH/"/

IF (SCFP .EQ. ' ') THEN
  SCFP = 'NLAC3D.SCR'
  ISCFP = 10
ELSE
  DO 100 K = 1, 40
    IF (SCFP(K:K) .EQ. ' ') THEN
      ISCFP = K - 1
    GO TO 110
  ENDIF
100  CONTINUE
110  CONTINUE
ENDIF

SCI = SLASH//SCFP(1:ISCFP)//.SCI
SC2 = SLASH//SCFP(1:ISCFP)//.SC2
SC3 = SLASH//SCFP(1:ISCFP)//.SC3
SC4 = SLASH//SCFP(1:ISCFP)//.SC4
SC6 = SLASH//SCFP(1:ISCFP)//.SC6
SC7 = SLASH//SCFP(1:ISCFP)//.SC7
PLT = SLASH//SCFP(1:ISCFP)//.PLT
IHT = SLASH//SCFP(1:ISCFP)//.IHT
IH1 = SLASH//SCFP(1:ISCFP)//.IH1
IH2 = SLASH//SCFP(1:ISCFP)//.IH2
IH3 = SLASH//SCFP(1:ISCFP)//.IH3
IH4 = SLASH//SCFP(1:ISCFP)//.IH4
IH5 = SLASH//SCFP(1:ISCFP)//.IH5
IH6 = SLASH//SCFP(1:ISCFP)//.IH6
IH7 = SLASH//SCFP(1:ISCFP)//.IH7
IH8 = SLASH//SCFP(1:ISCFP)//.IH8
IH9 = SLASH//SCFP(1:ISCFP)//.IH9
IHA = SLASH//SCFP(1:ISCFP)//.IHA
IHB = SLASH//SCFP(1:ISCFP)//.IHB
IHC = SLASH//SCFP(1:ISCFP)//.IHC
IHD = SLASH//SCFP(1:ISCFP)//.IHD
IHE = SLASH//SCFP(1:ISCFP)//.IHE
IHF = SLASH//SCFP(1:ISCFP)//.IHF
STY = SLASH//SCFP(1:ISCFP)//.STY
ST1 = SLASH//SCFP(1:ISCFP)//.ST1
ST2 = SLASH//SCFP(1:ISCFP)//.ST2
AFK = SLASH//SCFP(1:ISCFP)//.AFK
SPG = SLASH//SCFP(1:ISCFP)//.SPG
SG1 = SLASH//SCFP(1:ISCFP)//.SG1
SG2 = SLASH//SCFP(1:ISCFP)//.SG2
CK1 = SLASH//SCFP(1:ISCFP)//.CK1
CL0PF = 0
IF (OFH .NE. ' ') THEN
  OUT = SLASH/OFH
  CRK = SLASH/OFH
ELSE
  DO 120 K = 1, 40
    IF (OFP(K:K) .EQ. ' ') THEN
      IOFP = K - 1
      GO TO 130
    ENDIF
  120 CONTINUE
  OUT = SLASH/OFPP(1:IOFP)//'.OUT'
  CRK = SLASH/OFPP(1:IOFP)//'.CRK'
ENDIF

C INQUIRE(FILE=SC1, EXIST=THERE)
IF (THERE) THEN
  VOL = ' '
ELSE
  VOL = VOLSER
ENDIF
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 240, 'VOLSER', VOL, 'TRK'
  1, 20, 'SECOND', 40)
OPEN(I, STATUS='UNKNOWN', FILE=SC1)
C INQUIRE(FILE=SC2, EXIST=THERE)
IF (THERE) THEN
  VOL = ' '
ELSE
  VOL = VOLSER
ENDIF
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 240, 'VOLSER', VOL, 'TRK'
  1, 20, 'SECOND', 40)
OPEN(2, STATUS='UNKNOWN', FILE=SC2)
C INQUIRE(FILE=SC3, EXIST=THERE)
IF (THERE) THEN
  VOL = ' '
ELSE
  VOL = VOLSER
ENDIF
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 240, 'VOLSER', VOL, 'TRK'
  1, 20, 'SECOND', 40)
OPEN(3, STATUS='UNKNOWN', FILE=SC3)
C INQUIRE(FILE=SC4, EXIST=THERE)
IF (THERE) THEN
  VOL = ' '
ELSE
  VOL = VOLSER
ENDIF
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 240, 'VOLSER', VOL, 'TRK'
  1, 20, 'SECOND', 40)
OPEN(4, STATUS='UNKNOWN', FILE=SC4)
C INQUIRE(FILE=SC5, EXIST=THERE)
IF (THERE) THEN
  VOL = ' '
ELSE
  VOL = VOLSER
ENDIF
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 130, 'VOLSER', VOL, 'TRK'
1  , 20, 'SECOND', 40)
OPEN(16, STATUS='UNKNOWN', FILE=SC5)
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 130, 'VOLSER', VOL, 'TRK'
1  , 20, 'SECOND', 40)
OPEN(14, STATUS='UNKNOWN', FILE=SC6)
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 130, 'VOLSER', VOL, 'TRK'
1  , 20, 'SECOND', 40)
OPEN(16, STATUS='UNKNOWN', FILE=SC7)
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 130, 'VOLSER', VOL, 'TRK'
1  , 20, 'SECOND', 40)
OPEN(17, STATUS='UNKNOWN', FILE=PLT)
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 137, 'VOLSER', VOL, 'TRK'
1  , 20, 'SECOND', 40)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
OPEN(13, STATUS='OLD', FILE=OUT)
ICOUNT = 0
CONTINUE
READ (13, '(*80)') , END=150
ICOUNT = ICOUNT + 1
GO TO 140
CONTINUE
BACKSPACE 13
WRITE ('*150') 'OUT ICOUNT', ICOUNT
ENDIF
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1  , 20, 'SECOND', 40)
OPEN(13, STATUS='UNKNOWN', FILE=OUT)
ENDIF
CALL FILEIHF (I, 'RECFM', 'FB', 'LRECL', 130, 'VOLSER', VOL, 'TRK'
1  , 20, 'SECOND', 40)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
OPEN(20, STATUS='OLD', FILE=IST)
160 CONTINUE
READ (20, '(A80)', END=170)
GO TO 160
170 CONTINUE
BACKSPACE 20
ELSE
CLOSE(20, STATUS='UNKNOWN', FILE=IST)
ENDIF
C
INQUIRE(FILE=IST, EXIST=THERE)
IF (THERE) THEN
VOL = ' '
ELSE
VOL = VOLSER
ENDIF
CALL FILEINFO (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1, 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
OPEN(21, STATUS='OLD', FILE=IST)
180 CONTINUE
READ (21, '(A80)'), END=190)
GO TO 180
190 CONTINUE
BACKSPACE 21
ELSE
CLOSE(21, STATUS='UNKNOWN', FILE=IST)
ENDIF
C
INQUIRE(FILE=IST, EXIST=THERE)
IF (THERE) THEN
VOL = ' '
ELSE
VOL = VOLSER
ENDIF
CALL FILEINFO (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1, 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
OPEN(22, STATUS='OLD', FILE=IST)
200 CONTINUE
READ (22, '(A80)'), END=210)
GO TO 200
210 CONTINUE
BACKSPACE 22
ELSE
CLOSE(22, STATUS='UNKNOWN', FILE=IST)
ENDIF
C
INQUIRE(FILE=IST, EXIST=THERE)
IF (THERE) THEN
VOL = ' '
ELSE
VOL = VOLSER
ENDIF
CALL FILEINFO (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1, 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
OPEN(23, STATUS='OLD', FILE=IST)
220 CONTINUE
READ (23, '(A80)'), END=230)
GO TO 220

CONTINUE
BACKSPACE 23
ELSE
OPEN(23, STATUS='UNKNOWN', FILE=IN3)
ENDIF
C INQUIRE(FILE=IN4, EXIST=THERE)
IF (THERE) THEN
VOL ='
ELSE
VOL = VOLSER
ENDIF
CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1, 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
OPEN(24, STATUS='OLD', FILE=IN4)
CONTINUE
READ (24, '(A80)', END=260)
go to 240
C INQUIRE(FILE=IN4, EXIST=THERE)
IF (THERE) THEN
VOL ='
ELSE
VOL = VOLSER
ENDIF
CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1, 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
OPEN(26, STATUS='OLD', FILE=IN5)
CONTINUE
READ (26, '(A80)', END=290)
go to 260
C INQUIRE(FILE=IN5, EXIST=THERE)
IF (THERE) THEN
VOL ='
ELSE
VOL = VOLSER
ENDIF
CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1, 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
OPEN(26, STATUS='OLD', FILE=IN6)
CONTINUE
READ (26, '(A80)', END=290)
go to 280
C INQUIRE(FILE=IN6, EXIST=THERE)
IF (THERE) THEN
VOL ='
ELSE
VOL = VOLSER
ENDIF
CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1, 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
OPEN(26, STATUS='OLD', FILE=IN6)
C  
INQUIRE(FILE=IN7, EXIST=THERE)  
IF (THERE) THEN  
  VOL = ' '  
ELSE  
  VOL = VOLSER  
ENDIF  
CALL FILEINF (1, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK',  
1 , 40, 'SECOND', 80)  
IF (IFLAG3.EQ.1 .AND. THERE) THEN  
  OPEN(27, STATUS='OLD', FILE=IN7)  
  CONTINUE  
  READ (27, '(*A80)', END=310)  
  GO TO 300  
310  CONTINUE  
BACKSPACE 27  
ELSE  
  OPEN(27, STATUS='UNKNOWN', FILE=IN7)  
ENDIF  
C  
INQUIRE(FILE=IN8, EXIST=THERE)  
IF (THERE) THEN  
  VOL = ' '  
ELSE  
  VOL = VOLSER  
ENDIF  
CALL FILEINF (1, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK',  
1 , 40, 'SECOND', 80)  
IF (IFLAG3.EQ.1 .AND. THERE) THEN  
  OPEN(28, STATUS='OLD', FILE=IN8)  
  CONTINUE  
  READ (28, '(*A80)', END=330)  
  GO TO 320  
330  CONTINUE  
BACKSPACE 28  
ELSE  
  OPEN(28, STATUS='UNKNOWN', FILE=IN8)  
ENDIF  
C  
INQUIRE(FILE=IN9, EXIST=THERE)  
IF (THERE) THEN  
  VOL = ' '  
ELSE  
  VOL = VOLSER  
ENDIF  
CALL FILEINF (1, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK',  
1 , 40, 'SECOND', 80)  
IF (IFLAG3.EQ.1 .AND. THERE) THEN  
  OPEN(29, STATUS='OLD', FILE=IN9)  
  CONTINUE  
  READ (29, '(*A80)', END=360)  
  GO TO 360  
360  CONTINUE  
BACKSPACE 29  
ELSE  
  OPEN(29, STATUS='UNKNOWN', FILE=IN9)  
ENDIF  
C  
INQUIRE(FILE=IN4, EXIST=THERE)  
IF (THERE) THEN  
  VOL = ' '  
ELSE  
  VOL = VOLSER  
ENDIF
CALL FILEIF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
  1 , 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
  OPEN(30, STATUS='OLD', FILE=INA)
360 CONTINUE
  READ (30, '(A80)', END=370)
  GO TO 360
370 CONTINUE
  BACKSPACE 30
ELSE
  OPEN(30, STATUS='UNKNOWN', FILE=INA)
ENDIF

C
IQQUIRE(FILE=INB, EXIST=THERE)
IF (THERE) THEN
  VOL = ' ';
ELSE
  VOL = VOLSER
ENDIF
CALL FILEIF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
  1 , 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
  OPEN(31, STATUS='OLD', FILE=INB)
380 CONTINUE
  READ (31, '(A80)', END=390)
  GO TO 380
390 CONTINUE
  BACKSPACE 31:
ELSE
  OPEN(31, STATUS='UNKNOWN', FILE=INB)
ENDIF

C
IQQUIRE(FILE=IBC, EXIST=THERE)
IF (THERE) THEN
  VOL = ' ';
ELSE
  VOL = VOLSER
ENDIF
CALL FILEIF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
  1 , 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
  OPEN(32, STATUS='OLD', FILE=IBC)
400 CONTINUE
  READ (32, '(A80)', END=410)
  GO TO 400
410 CONTINUE
  BACKSPACE 32:
ELSE
  OPEN(32, STATUS='UNKNOWN', FILE=IBC)
ENDIF

C
IQQUIRE(FILE=IND, EXIST=THERE)
IF (THERE) THEN
  VOL = ' ';
ELSE
  VOL = VOLSER
ENDIF
CALL FILEIF (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
  1 , 40, 'SECOND', 50)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
  OPEN(33, STATUS='OLD', FILE=IND)
420 CONTINUE
READ (33, '(A80)', END=430)
GO TO 420
430 CONTINUE
BACKSPACE 33
ELSE
OPEN (33, STATUS='UNKNOWN', FILE=IND)
ENDIF
C
INQUIRE(FILE=INE, EXIST= THERE)
IF (THERE) THEN
VOL = ' '
ELSE
VOL = VOLSER
ENDIF
CALL FILEINFO (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1 , 40, 'RECORD', 80)
IF (IFLAG5. EQ.1 .AND. THERE) THEN
OPEN(34, STATUS='OLD', FILE=INE)
ENDIF
READ (34, '(A80)', END=450)
GO TO 440
450 CONTINUE
BACKSPACE 34
ELSE
OPEN(34, STATUS='UNKNOWN', FILE=INE)
ENDIF
C
INQUIRE(FILE=INF, EXIST= THERE)
IF (THERE) THEN
VOL = ' '
ELSE
VOL = VOLSER
ENDIF
CALL FILEINFO (I, 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1 , 40, 'RECORD', 80)
IF (IFLAG5. EQ.1 .AND. THERE) THEN
OPEN(35, STATUS='OLD', FILE=INF)
ENDIF
READ (35, '(A80)', END=470)
GO TO 460
470 CONTINUE
BACKSPACE 35
ELSE
OPEN(35, STATUS='UNKNOWN', FILE=INF)
ENDIF
C
INQUIRE(FILE=STY, EXIST= THERE)
IF (THERE) THEN
VOL = ' '
ELSE
VOL = VOLSER
ENDIF
CALL FILEINFO (I, 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
1 , 100, 'RECORD', 100, 'BSIZE', 32760)
OPEN(30, STATUS='UNKNOWN', FILE=STY)
C
INQUIRE(FILE=ST1, EXIST= THERE)
IF (THERE) THEN
VOL = ' '
ELSE
VOL = VOLSER
ENDIF
CALL FILEINFO (I, 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
1  ,  100, 'SECOND', 100, 'BLKSIZE', 32760)
OPEN(60, STATUS='UNKNOWN', FILE=ST1)
C
INQUIRE(FILE=ST2, EXIST='THERE')
IF (THERE) THEN
  VOL = ''
ELSE
  VOL = VOLSER
ENDIF
CALL FILEINF (I , 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
1  ,  100, 'SECOND', 100, 'BLKSIZE', 32760)
OPEN(61, STATUS='UNKNOWN', FILE=ST2)
C
INQUIRE(FILE=AFK, EXIST='THERE')
IF (THERE) THEN
  VOL = ''
ELSE
  VOL = VOLSER
ENDIF
CALL FILEINF (I , 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
1  ,  100, 'SECOND', 100, 'BLKSIZE', 32760)
OPEN(62, STATUS='UNKNOWN', FILE=AFK)
C
INQUIRE(FILE=SPG, EXIST='THERE')
IF (THERE) THEN
  VOL = ''
ELSE
  VOL = VOLSER
ENDIF
CALL FILEINF (I , 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
1  ,  100, 'SECOND', 100, 'BLKSIZE', 32760)
OPEN(80, STATUS='UNKNOWN', FILE=SPG)
C
INQUIRE(FILE=SG1, EXIST='THERE')
IF (THERE) THEN
  VOL = ''
ELSE
  VOL = VOLSER
ENDIF
CALL FILEINF (I , 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
1  ,  100, 'SECOND', 100, 'BLKSIZE', 32760)
OPEN(70, STATUS='UNKNOWN', FILE=SG1)
C
INQUIRE(FILE=SG2, EXIST='THERE')
IF (THERE) THEN
  VOL = ''
ELSE
  VOL = VOLSER
ENDIF
CALL FILEINF (I , 'RECFM', 'VS', 'LRECL', -1, 'VOLSER', VOL, 'TRK'
1  ,  100, 'SECOND', 100, 'BLKSIZE', 32760)
OPEN(71, STATUS='UNKNOWN', FILE=SG2)
C
INQUIRE(FILE=CK1, EXIST='THERE')
IF (THERE) THEN
  VOL = ''
ELSE
  VOL = VOLSER
ENDIF
CALL FILEINF (I , 'RECFM', 'FB', 'LRECL', 80, 'VOLSER', VOL, 'TRK'
1  ,  40, 'SECOND', 60)
IF (IFLAG3.EQ.1 .AND. THERE) THEN
  OPEN(72, STATUS='OLD', FILE=CK1)
480 CONTINUE
READ (72, '(A80)', END=490)
GO TO 480
490 CONTINUE
BACKSPACE 72
ELSE
OPEN (72, STATUS='UNKNOWN', FILE=CK)
ENDIF
C
INQUIRE(FILE=CRK, EXIST='THERE')
IF ('THERE') THEN
VOL = ' '
ELSE
VOL = VOLSER
ENDIF
CALL FILEINF (I, 'RECFM', 'FB', 'LRECL', 137, 'VOLSER', VOL, 'TRK',
1, 100, 'SECOND', 100)
IF (IFLAG3.EQ.1 .AND. 'THERE') THEN
OPEN (40, STATUS='OLD', FILE=CRK)
ELSE
READ (40, '(A80)', END=610)
GO TO 600
ENDIF
510 CONTINUE
BACKSPACE 40
ELSE
OPEN (40, STATUS='UNKNOWN', FILE=CRK)
ENDIF
C
WRITE(*,*) 'EXITING FILES'
RETURN
C
ENTRY CFILE
C
IF (KEEP.EQ.0 .OR. IDEL.EQ.1) THEN
STAT = 'DELETE'
ELSE
STAT = 'KEEP'
ENDIF
C
CLOSE(1, STATUS='STAT')
CLOSE(2, STATUS='STAT')
CLOSE(3, STATUS='STAT')
CLOSE(4, STATUS='STAT')
CLOSE(16, STATUS='STAT')
CLOSE(14, STATUS='STAT')
CLOSE(15, STATUS='STAT')
CLOSE(60, STATUS='STAT')
CLOSE(60, STATUS='STAT')
CLOSE(61, STATUS='STAT')
CLOSE(62, STATUS='STAT')
CLOSE(60, STATUS='STAT')
CLOSE(70, STATUS='STAT')
CLOSE(71, STATUS='STAT')
CLOSE(17, STATUS='KEEP')
CLOSE(13, STATUS='KEEP')
CLOSE(40, STATUS='KEEP')
CLOSE(72, STATUS='KEEP')
CLOSE(18, STATUS='KEEP')
INQUIRE(FILE=INT, EXIST='THERE')
IF ('THERE') THEN
CLOSE(20, STATUS='KEEP')
ELSE
    CLOSE(20, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IH1, EXIST=THERE)
IF (THERE) THEN
    CLOSE(21, STATUS='KEEP')
ELSE
    CLOSE(21, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IH2, EXIST=THERE)
IF (THERE) THEN
    CLOSE(22, STATUS='KEEP')
ELSE
    CLOSE(22, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IH3, EXIST=THERE)
IF (THERE) THEN
    CLOSE(23, STATUS='KEEP')
ELSE
    CLOSE(23, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IH4, EXIST=THERE)
IF (THERE) THEN
    CLOSE(24, STATUS='KEEP')
ELSE
    CLOSE(24, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IH5, EXIST=THERE)
IF (THERE) THEN
    CLOSE(25, STATUS='KEEP')
ELSE
    CLOSE(25, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IH6, EXIST=THERE)
IF (THERE) THEN
    CLOSE(26, STATUS='KEEP')
ELSE
    CLOSE(26, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IH7, EXIST=THERE)
IF (THERE) THEN
    CLOSE(27, STATUS='KEEP')
ELSE
    CLOSE(27, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IH8, EXIST=THERE)
IF (THERE) THEN
    CLOSE(28, STATUS='KEEP')
ELSE
    CLOSE(28, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IH9, EXIST=THERE)
IF (THERE) THEN
...
CLOSE(29, STATUS='KEEP')
ELSE
CLOSE(29, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IHA, EXIST='THERE')
IF (THERE) THEN
CLOSE(30, STATUS='KEEP')
ELSE
CLOSE(30, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IHB, EXIST='THERE')
IF (THERE) THEN
CLOSE(31, STATUS='KEEP')
ELSE
CLOSE(31, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=ICH, EXIST='THERE')
IF (THERE) THEN
CLOSE(32, STATUS='KEEP')
ELSE
CLOSE(32, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IID, EXIST='THERE')
IF (THERE) THEN
CLOSE(33, STATUS='KEEP')
ELSE
CLOSE(33, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IIE, EXIST='THERE')
IF (THERE) THEN
CLOSE(34, STATUS='KEEP')
ELSE
CLOSE(34, STATUS='DELETE')
ENDIF
C
INQUIRE(FILE=IFF, EXIST='THERE')
IF (THERE) THEN
CLOSE(35, STATUS='KEEP')
ELSE
CLOSE(35, STATUS='DELETE')
ENDIF
C
RETURN
C
ENTRY ERNFIL
OPEN(18, ACCESS='DIRECT', RECL=80, FILE='/CERAMA.WLAC3D.ERR')
RETURN
C
END
C
INCLUDE (PROCESS)
SUBROUTINE INITIAL
IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: REUMB=100:10,FORMAT=900:10
C...SWITCHES:
C******************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBRoutines OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C*****************************************************************************
REAL*4 XI, ETA, SI, FMAG, DMAG, SHELLZ
CHARACTER*80 GTITLE
COMM0H/MAIHi/U(60000), REJ(60000)
COMM0H/MAIH4/RE(60000)
COMM0H/INPUT7/RE(60000), RX(60000), RZ(60000)
COMM0H/INPUT4/ISPBC(10000)
COMM0H/IREP/IREP(12000), LREP(12000)
COMM0H/IINPUT/T/INFLU(6, 6000)
COMM0H/SKTR2/SHULLZ(3, 10000)
COMM0H/SAFE2/R(60000), IDOF(60000), JDIAG(60000)
COMM0H/SAFE4/INDOF(60000), IDDF(60000)
COMM0H/PATH/HISTX(10, 30), HISTY(10,30), IPATH(10)
COMM0H/HPCD/GUSEPH(40000), ALAM(40000), MPCDOF(40000),
$ MPCADR(3,6000), MPCC, MPCSTR, MAIMPC
COMM0H/GUTP2/IDEA(6000), IDMUD(10000)
COMM0H/ICKFIP/ICKFGO, ICKHTR
ICKFGO = 0
ICKHTR = 1
C
C MAXMPC = 5000
C
C DO IJ = 1, 10
IPATH(10) = 0
DO KL = 1, 30
HISTX(IJ,KL) = 0.0
HISTY(IJ,KL) = 0.0
C
END DO
END DO
IPATH(1) = 2
IPATH(2) = 2
IPATH(3) = 2
HISTX(2,1) = 1.0
HISTY(2,1) = 1.0
HISTX(2,2) = 1.0
HISTY(2,2) = 1.0
HISTX(3,1) = 0.0
HISTY(3,1) = 0.0
HISTX(3,2) = 1.0
HISTY(3,2) = 1.0
C
DO KP = 1, 40000
MPCDOF(KP) = 0
GUSEPH(KP) = 0.0
ALAM(KP) = 0.0
END DO
C
END DO
C DO KB = 1, 5000
MPCADR(1,KB) = 0
MPCADR(2,KB) = 0
END DO
C
C DO X1 = 1, 10000
IZOD(1) = 0
ISPBC(1) = 0
SHELLZ(1,X1) = 0
SHELLZ(2,X1) = 0
SHELLZ(3,X1) = 0
END DO
C
C DO X1 = 1, 6000
IDEA(X1) = 0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
DO K2 = 1, 6
   IHDEL(K2,K1) = 0.
END DO
END DO

DO K1 = 1, 60000
   IDOF(K1) = 0
   JDIAGCK1) = 0
   RCK1) = 0.
   U(K1) = 0.
   IRDOF(K1) = 0.
   IUDOF(K1) = 0.
   Rx(K1) = 0.
   RY(K1) = 0.
   RZ(K1) = 0.
   RE(K1) = 0.
   RE1(K1) = 0.
END DO

DO K1 = 1, 120000
   IREP(K1) = 0
   LREP(K1) = 0
END DO
RETURN
END

C===================================================================
C
C INCLUDE (PROCESS)
SUBROUTINE INPUT(IOUT,IERROR)

PROGRAM:
C INPUT is the main input routine for HLAC3D. This program reads
C each record of the input file and extracts the command by
C calling subroutine COMPDD. The extracted commands are then
C processed appropriately.
C
C ENTRY:
C IOUT = output device number
C
C RETURN:
C IDOF = is the array containing the degree of freedom
C information for each node.
C =1; degree of freedom is constrained (zero displ.)
C =0; degree of freedom is unconstrained
C =-1; degree of freedom has a prescribed value and
C is constrained (non-zero displacement)
C===================================================================

IMPLICIT REAL*8 (A-H,0-Z)
C...SWITCHES: REHUMB=100:10,FORHAT=900:10
C...SWITCHES: REHUMB=100:10,FORMAT=900:10
C...SWITCHES:
C===================================================================
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C EBORAS  COMPFO  IDISP  IDCHS  CEPROP  TOLOAD
C SULOAD  GRAPHS  IDMATR  IMANDY  ELPREP  IDCHS
C ELEHEH  IDFREE  IDOUT  FILES  CASES  ERRORS
C OUT11  C
C*******************************************************************************
REAL*8 LX,LY
REAL*4 XY2,THICK
CHARACTER*80 BUFFER,BUFF,TITLE
CHARACTER*40 SCFP,OFP,OFH,VOLSER,UHIVER
CHARACTER*6 COMM
COMM0H/IBPUT9/THICK(9),IFLAG
COMM0H/MAIHl/U(60000),REL(60000)
COMM0H/IHPUT2/NP(20,6000)
COMM0H/IHPUT3/YT(3,10000)
C
C ---- Common IHPUT7 is used with a different organization in module
C SAFE. IT IS USED HERE FOR THE PURPOSE OF SAVING STORAGE FOR
C holding some temporary parameters. Its size should be
C consistent with its specification in module safe
C
COMM0H/IHPUT7/HHELEH(360000)
C
C nnelem = number of nodes for the element
C
COMM0H/IHPUT8/NODES, BELEM,HELEM, HNDF, HLIFD, HHIH, HNF, HNF2, IFLAG1, IFLAG2, IDIK, 1
BIODE, BCOLOR, BFREE
COMM0H/IBCR11/FRACTD,O) ,BLIHC1O) ,LDCOBT, LCPTR
COMM0H/IBPUTB/FAC, FACHEW,FACLDW,FACHIG,EHRGl, HDIVER, ISTOP
COMM0H/IBUTC/TITLE
COMM0H/IBPUTC/ISPB(10000)
COMM0H/IBPUTD/IPLA03,J0INTR, IFPLOT
COMM0H/BOUND1/BOUVES
COMM0H/BOUND2/IXZ(6),IXZ(6),IXZ(6),IXZ(6)
COMM0H/IHPUTI/INTFACE(600)
COMM0H/FILEK/KEEP, IDEL, SCFP, OFP, OFP, VOLSER
COMM0H/COND2/COMB, BUFFER, BUFF
COMM0H/POINTS/LX(4, 6), LY(4, 6)
COMM0H/SAFE2/R(60000), IDOF(60000), IDIAG(60000)
COMM0H/MPCS/CDEFPMP(40000), ALAMB(40000), MCPDOF(40000), $MPCDR(2,5000), MPC, MPCST, MAIPMC
COMM0H/LPROP/FRSPR(25,16)
COMM0H/ICESP0/IICETF0,IICETR
COMM0H/IICEMP/IICEMP
DIMENSION DUMMY(6),M(20),LAMX(6000,2)
CALL EBORAS
C
C ---- read one line of the input file and store in BUFFER
C
C C .... initialize the output vector for elements/nodes
C
LDEVII = 11
ECURVE = 0
LDCOBT = 0
RI = 0
MAIPMC = 5000
IFLAG7 = 0
MPC = 0
MPCST = 0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
**Variables and Code Lines**

```plaintext
!BHCTR = 1
IFLAG3 = 0
DO IK = 1, 5000
   LAMIHA(IK,1) = 0.0
   LAMIHA(IK,2) = 0.0
END DO
110 CONTINUE
READ (LDEV11, '(A80)', END=300) BUFFER
C
C     extract the first six characters of each command in the BUFFER
C     and place all variables associated with each command in the
C     internal file BUFF.
C
120 CONTINUE
CALL CUMP80 (N, NVAR)
C
130 CONTINUE
IF (N .EQ. 0) THEN
   ASSIGN 110 TO NEXT
ELSE
   ASSIGN 120 TO NEXT
ENDIF
C
IF (COMM .EQ. 'TITLE') THEN
   GO TO 150
ELSE IF (COMM .EQ. 'COMHEN') THEN !  COMMENT LINE
   GO TO 110
ELSE IF (COMM.EQ.'COORDI' .OR. COMM.EQ.'NODES' )  THEN
   GO TO 170
ELSE IF (COMM.EQ.'NODE* .OR. COMM.EQ.'JOINTS') THEN
   GO TO 170
ELSE IF (COMM.EQ. 'MEMBER' .OR. COMM.EQ. 'INCIDE') THEN
   GO TO 210
ELSE IF (COMM .EQ. 'COSHEC') THEN
   GO TO 210
ELSE IF (COMM .EQ. 'DISPLA') THEN
   CALL IODISP (IDOF, NDIF, IDIM, N, ICOMM, LDEV11, IOUT)
   IF (ICOMM .EQ. 1) THEN
      GO TO 130
   ELSE IF (ICOMM .EQ. 2) THEN
      GO TO 300
   ELSE
      GO TO 110
   ENDIF
ELSE IF (COMM .EQ. 'CONSEC') THEN
   GO TO 210
ELSE IF (COMM .EQ. 'DISPLA') THEN
   CALL IODISP (IDOF, NDIF, IDIM, N, ICOMM, LDEV11, IOUT)
   IF (ICOMM .EQ. 1) THEN
      GO TO 130
   ELSE IF (ICOMM .EQ. 2) THEN
      GO TO 300
   ELSE
      GO TO 110
   ENDIF
ELSE IF (COMM .EQ. 'COFSTR') THEN
   CALL IOCONST (LAMINA, IOUT, IERROR, ICOMM)
   IF (IFLAG3 .EQ. 0) CALL CSPROP (LAMINA, IOUT, IERROR)
   IF (ICOMM .EQ. 1) THEN
      GO TO 130
   ELSE IF (ICOMM .EQ. 2) THEN
      GO TO 300
   ELSE
      GO TO 110
   ENDIF
ELSE IF (COMM.EQ.'LOAD' .OR. COMM.EQ.'LOADS') THEN
   CALL IOLOAD (IDIM, N, NDIF, ICOMM, LDEV11, IOUT)
   IF (ICOMM .EQ. 1) THEN
      GO TO 130
   ELSE IF (ICOMM .EQ. 2) THEN
      GO TO 300
   ELSE
      GO TO 110
   ENDIF
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ELSE IF (COMM.EQ.'SURFAC' .OR. COMM.EQ.'SU.LOD') THEN
    CALL SULOAD (IDIM, H, HDDF, ICOMM, LDEV11, IOUT)
    IF (ICOMM .EQ. 1) THEN
        GO TO 130
    ELSE IF (ICOMM .EQ. 2) THEN
        GO TO 300
    ELSE
        GO TO 110
ENDIF
ELSE IF (COMM .EQ. 'GRAPHIC') THEN
    READ (BUFF, *, END=290) IPLOT
    CALL GRAPHIC (H, ICOMM, LDEV11, IOUT)
    IF (ICOMM .EQ. 1) THEN
        GO TO 130
    ELSE IF (ICOMM .EQ. 2) THEN
        GO TO 300
    ELSE
        GO TO 110
ENDIF
ENDIF
ELSE IF (COMM .EQ. 'MATERI' .OR. COMM.EQ.'MAT') THEN
    CALL IMATR (H, ICOMM, LDEV11, IOUT)
    IF (ICOMM .EQ. 1) THEN
        GO TO 130
    ELSE IF (ICOMM .EQ. 2) THEN
        GO TO 300
    ELSE
        GO TO 110
ENDIF
ELSE IF (COMM .EQ. 'STOP') THEN
    READ (BUFF, *, END=290) ISTOP
    ELSE IF (COMM .EQ. 'LINEAR') THEN
        IFLAG1 = 0
    ELSE IF (COMM .EQ. 'NONLIN') THEN
        IFLAG1 = 1
    ELSE IF (COMM .EQ. 'NOSYM') THEN
        IFLAG1 = 1
    ELSE IF (COMM .EQ. 'SYM') THEN
        IFLAG1 = 0
    ELSE IF (COMM .EQ. 'BCPREP') THEN
        CALL IXBARP (PROPER, IOUT, IERROR, MATHUM)
        IF (IFLAG3 .EQ. 0) CALL HLPROP (IOUT, IERROR, MATHUM)
    ELSE IF (COMM.EQ. 'HISTOR' .OR. COMM.EQ.'HIST') THEN
        CALL IHIST (IOUT, IERROR)
    ELSE IF (COMM.EQ. 'ELEMEH' .OR. COMM.EQ.'ELEM') THEN
        CALL ELEMEH (H, ICOMM, LDEV11, IOUT, HHELEM)
        IF (ICOMM .EQ. 1) THEN
            GO TO 130
        ELSE IF (ICOMM .EQ. 2) THEN
            GO TO 300
        ELSE
            GO TO 110
ENDIF
ELSE IF (COMM.EQ. 'FREEBO') THEN
    CALL IOFREE (H, ICOMM, LDEV11, IOUT)
    IF (ICOMM .EQ. 1) THEN
        GO TO 130
    ELSE IF (ICOMM .EQ. 2) THEN
        GO TO 300
    ELSE
        GO TO 110
ENDIF
ELSE IF (COMM .EQ. 'DINES') THEN
   READ (BUFF, *, END=290) IDIM
   IF (IDIM .EQ. 3) THEN
      ENDIF
   ELSE
      ENDF = IDIM
   ENDIF
ELSE IF (COMM .EQ. 'ITERA') THEN
   READ (BUFF, *, END=290) MBIT
ELSE IF (COMM .EQ. 'ICHREM') THEN
   LDCOHT = LDCOHT + 1
   READ (BUFF, *, END=290) HLIHCl(LDCOHT)
ELSE IF (COMM .EQ. 'ICHREI') THEN
   READ (BUFF, *, END=290) HLIBCl
ELSE IF (COMM .EQ. 'ICKHTR') THEN
   READ (BUFF, *, END=290) ICKHTR
ELSE IF (COMM .EQ. 'FRACTI' .OR. COMM .EQ. 'FRACT') THEN
   READ (BUFF, *, END=290) FRACT(LDCOHT)
ELSE IF (COMM .EQ. 'FACLOV') THEN
   FAC = FACLOV
ELSE IF (COMM .EQ. 'PAX') THEN
   READ (BUFF, *, END=290) LX(1,HCURVE)
ELSE IF (COMM .EQ. 'PAY') THEN
   READ (BUFF, *, END=290) LY(1,HCURVE)
ELSE IF (COMM .EQ. 'PBX') THEN
   READ (BUFF, *, END=290) LX(2,HCURVE)
ELSE IF (COMM .EQ. 'PBY') THEN
   READ (BUFF, *, END=290) LY(2,HCURVE)
ELSE IF (COMM .EQ. 'RAX') THEN
   READ (BUFF, *, END=290) LX(3,HCURVE)
ELSE IF (COMM .EQ. 'RAY') THEN
   READ (BUFF, *, END=290) LY(3,HCURVE)
ELSE IF (COMM .EQ. 'RBX') THEN
   READ (BUFF, *, END=290) LX(4,HCURVE)
ELSE IF (COMM .EQ. 'RBY') THEN
   READ (BUFF, *, END=290) LY(4,HCURVE)
ELSE IF (COMM .EQ. 'CURVE') THEN
   READ (BUFF, *, END=290) HCURVE
140 CONTINUE
   IF (COMM .EQ. 'PAX') THEN
      READ (BUFF, *, END=290) LX(1,HCURVE)
   ELSE IF (COMM .EQ. 'PAY') THEN
      READ (BUFF, *, END=290) LY(1,HCURVE)
   ELSE IF (COMM .EQ. 'PBX') THEN
      READ (BUFF, *, END=290) LX(2,HCURVE)
   ELSE IF (COMM .EQ. 'PBY') THEN
      READ (BUFF, *, END=290) LY(2,HCURVE)
   ELSE IF (COMM .EQ. 'RAX') THEN
      READ (BUFF, *, END=290) LX(3,HCURVE)
   ELSE IF (COMM .EQ. 'RAY') THEN
      READ (BUFF, *, END=290) LY(3,HCURVE)
   ELSE IF (COMM .EQ. 'RBX') THEN
      READ (BUFF, *, END=290) LX(4,HCURVE)
   ELSE IF (COMM .EQ. 'RBY') THEN
      READ (BUFF, *, END=290) LY(4,HCURVE)
   ELSE IF (COMM .EQ. 'CURVE') THEN
      READ (BUFF, *, END=290) HCURVE
   GO TO NEXT

BCURV = MAXO(BCURVE, BCURV)
ELSE IF (COMB .EQ. 'IZL') THEN
  READ (BUFF, *, END=290) XZL(BCURVE)
ELSE IF (COMB .EQ. 'YZB') THEN
  READ (BUFF, *, END=290) YZB(BCURVE)
ELSE IF (COMB .EQ. 'YFT') THEN
  READ (BUFF, *, END=290) YFT(BCURVE)
ELSE IF (COMB .EQ. 'SCFP') THEN
  READ (BUFF, *, END=290) SCFP
ELSE IF (COMB .EQ. 'OFP') THEN
  READ (BUFF, *, END=290) OFP
ELSE IF (COMB .EQ. 'OFP') THEN
  READ (BUFF, *, END=290) OFP
ELSE IF (COMB .EQ. 'VOLS') THEN
  READ (BUFF, *, END=290) VOLS
ELSE IF (COMB .EQ. 'KEEP') THEN
  KEEP = 1
ELSE IF (COMB .EQ. 'DEL' .OR. COMB .EQ. 'DELETE') THEN
  IDEL = 1
ELSE IF (COMB .EQ. 'BEGIN') THEN
  CALL FILES
ELSE IF (COMB .EQ. 'END') THEN
  CALL FILES
ELSE IF (COMB .EQ. 'SCRE') THEN
  Call SCRE
ELSE
  CALL ERRORS (2, 1, 'IOUTPUT')
ENDIF
GO TO NEXT
C changes 7/16
C --- read and generate the nodal coordinates
C
150 CONTINUE
  READ (BUFF, *, END=290) NUMBER
160 K = 1, NUMBER
  READ (LDEV11, '(ABC)') TITLE
  WRITE (TOUT, '(ABO)') TITLE
GO TO NEXT
160 CONTINUE
C ---- read the title cards from input file and echo into output files
C
170 CONTINUE
  I = 0
180 CONTINUE
  READ (LDEV11, *) K, (DUMMY(IDIR), IDIR = 1, IDIM), IHCR
C Set bit 10 of ISPB to 1 to indicate that the position of the
C node has been defined.
C
ISPB(K) = IBSET(ISPB(K),10)
XYZ(1,K) = DUMMY(1)
XYZ(2,K) = DUMMY(2)
XYZ(3,K) = DUMMY(3)
C
I = I + 1
IF (IHCR .EQ. 0) GO TO 200
N = (X-K1)/IHCR
DX = (XYZ(1,K)-XYZ(1,K1))/N
DY = (XYZ(2,K)-XYZ(2,K1))/N

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ DZ = \frac{XYZ(3,3) - XYZ(3,K1)}{H} \]

\[ K2 = K - \text{INCR} \]

Do 190 \( J = K1, K2, \text{INCR} \)

\[ \text{ISPFB}(J) = \text{IBSET}(\text{ISPFB}(J),10) \]

\[ M1 = (J-K1)/\text{INCR} \]

\[ XYZ(1,J) = XYZ(1,K1) + M1*DX \]

\[ XYZ(2,J) = XYZ(2,K1) + M1*DY \]

\[ XYZ(3,J) = XYZ(3,K1) + M1*DZ \]

\[ I = I + 1 \]

190 CONTINUE

\[ I = I - 1 \]

200 CONTINUE

\[ K1 = K \]

IF (I .LT. \text{NBODES}) GO TO 180

GO TO BEXT

C ---- read and generate the element connectivity

C

210 CONTINUE

\[ I = 0 \]

READ (BUFF, *, END=290) \text{BELEM}

220 CONTINUE

READ (LDEV1, '(*200)', END=300) BUFFER

READ (BUFF, *) X

\[ \text{NBEL} = \text{BELEH}(X) \]

IF (\text{NBEL} .LE. 0) CALL \text{ERRORS}(11, 1, 'IHPUT ')

READ (BUFFER, *) X, (BOP(BODE,K), BODE = 1, BHEL), \text{INCR}

\[ I = I + 1 \]

C ---- set bit 20 of ISPFB to 1 to indicate that the connectivity for

the node has been defined.

C

\[ \text{ISPFB}(K) = \text{IBSET}(\text{ISPFB}(K),20) \]

IF (\text{INCR} .EQ. 0) THEN

\[ K1 = K \]

ELSE

\[ X2 = (X-K1)/\text{INCR} \]

DO 230 \text{NODE} = 1, \text{NBEL}

\[ M(\text{NODE}) = (\text{BOP}(\text{NODE},X) - \text{BOP}(\text{NODE},K))/X2 \]

230 CONTINUE

C

DO 250 \text{IELEM} = K1 + \text{INCR}, K - \text{INCR}

\[ I = I + 1 \]

\[ \text{IELEM1} = \text{IELEM} - \text{INCR} \]

\[ \text{ISPFB}(\text{IELEM}) = \text{IBSET}(\text{ISPFB}(\text{IELEM}),20) \]

DO 240 \text{NODE} = 1, \text{NBEL}

\[ \text{BOP}(\text{NODE},\text{IELEM}) = \text{BOP}(\text{NODE},\text{IELEM1}) + M(\text{NODE}) \]

240 CONTINUE

250 CONTINUE

ENDIF

IF (I .LT. \text{BELEM}) GO TO 220

GO TO BEXT

C ---- read and generate the interface nodes

C

260 CONTINUE

\[ I = 0 \]

READ (BUFF, *, END=290) \text{BIBODE}

270 CONTINUE

READ (LDEV1, *) X, \text{INCR}

IF (\text{INCR} .EQ. 0) THEN

\[ I = I + 1 \]
ISTFAC(I) = K
ELSE
ISTART = ISTFAC(I) + IBCR
IEND = K
DO 280 J = ISTART, IEND, IBCR
  I = I + 1
  ISTFAC(I) = J
280 CONTINUE
ENDIF
IF (I .LT. NNODES) GO TO 270
GO TO NEXT
C
290 CONTINUE
CALL ERRORS (3, 1, 'INPUT ')  
300 CONTINUE
CALL OUT11 (IDOUT)  
RETURN
END
C
=== INCLUDE (PROCESS) ===
SUBROUTINE IODISP(IDOF, BBDF, IDIM, N, ICOMM, LDEV11, IDOUT)  
=== PROGRAM: ===
IODISP reads and stores the nodal displacements in either the global or the local coordinate system.
C
C CONSTRICTION:
C 0; D.O.F. restrained with zero displacement or rot.
C 1; D.O.F. is free
C -1; D.O.F. is restrained with non zero displacements
C 1; Buffer contains at least one additional command
C 0; buffer contains no additional commands
C 1; Buffer contains at least one additional command
C 0; read the next input line
C 1; The current command was not resolved by IODIS.
C 1; Calling routine should process this command
C 2; end of file is reached, do not try to read any more lines.
C
C Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RESUME=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C COMPRA IDCORD ERRORS
C***********************************************************************
LOGICAL*4 ISP
CHARACTER*80 BUFFER,BUFF
CHARACTER*8 COMM
REAL*4 XAX,YAX,ZAX,XAXIS,YAXIS
REAL*4 XAXR,YAXR,ZAXR
COMMON/MAIN1/U(60000),RE1(60000)
COMMON/INPUTD/XAXIS(4,10000),YAXIS(4,10000)
COMMON/ISPUTE/ISP1U(10000)
COMMON/COMP2/COMM,BUFFER,BUFF
COMMON/SAFE/IREDP(60000),IUDP(60000)
DIMENSION IDOP(1),D(6),IDO(6),ZAX(3),YAX(3)
DIMENSION XAXR(3),YAXR(3),ZAXR(3),IUD(6)
C
IDCORD = 0
IDCORD = 0
ICOMM = 0
ITIME = 0
ISTART = 0
IEND = 0
ISTR = 0
C
IF (N .NE. 0) GO TO 110
100 CONTINUE
READ (LDEV11,'(A80)',END=210) BUFFER
110 CONTINUE
CALL COMPRA (N, HVAR)
C
120 CONTINUE
IF (COMM.EQ.'NODE' .OR. COMM.EQ.'NODES') THEN
  IF (ITIME .EQ. 1)
    ASSIGN 130 TO NEXT
  GO TO 180
ENDIF
130 CONTINUE
READ (BUFF, *, END=160) ISTART
ITIME = 1
IEND = ISTART
INTR = 1
DO 140 K1 = 1, HNDF
  D(K1) = 0.
  IDO(K1) = 0
140 CONTINUE
C
C ---- IDCORD is the flag for the translation coordinate definition.
C = 0; a local coordinate system is not defined
C = 1; a local coordinate system is defined
C ---- IRCORD is the flag for the rotational coordinate definition.
C = 0; a local coordinate system is not defined
C = 1; a local coordinate system is defined
C
IDCORD = 0
IDCORD = 0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ELSE IF (COMM .EQ. 'TO') THEN
READ (BUFF, *, END=160) IEHD
ELSE IF (COMM .EQ. 'TY') THEN
READ (BUFF, *, END=160) ITR
ELSE IF (COMM .EQ. 'DI') THEN
READ (BUFF, *, END=160) D(1)
IF (D(1).EQ.0.) THEN
  ID0(1) = 1
ELSE
  ID0(1) = -1
ENDIF
ELSE IF (COMM .EQ. 'DY') THEN
READ (BUFF, *, END=160) D(2)
IF (D(2).EQ.0.) THEN
  ID0(2) = 1
ELSE
  ID0(2) = -1
ENDIF
ELSE IF (COMM .EQ. 'DZ') THEN
READ (BUFF, *, END=160) D(3)
IF (D(3).EQ.0.) THEN
  ID0(3) = 1
ELSE
  ID0(3) = -1
ENDIF
ELSE IF (COMM .EQ. 'RX') THEN
READ (BUFF, *, END=160) D(4)
IF (D(4).EQ.0.) THEN
  ID0(4) = 1
ELSE
  ID0(4) = -1
ENDIF
ELSE IF (COMM .EQ. 'RY') THEN
READ (BUFF, *, END=160) D(5)
IF (D(5).EQ.0.) THEN
  ID0(5) = 1
ELSE
  ID0(5) = -1
ENDIF
ELSE IF (COMM .EQ. 'RZ') THEN
READ (BUFF, *, END=160) D(6)
IF (D(6).EQ.0.) THEN
  ID0(6) = 1
ELSE
  ID0(6) = -1
ENDIF
ELSE IF (COMM .EQ. 'EDI') THEN
READ (BUFF, *, END=160) IUD(1)
ELSE IF (COMM .EQ. 'EDY') THEN
READ (BUFF, *, END=160) IUD(2)
ELSE IF (COMM .EQ. 'EDZ') THEN
READ (BUFF, *, END=160) IUD(3)
ELSE IF (COMM .EQ. 'EBI') THEN
READ (BUFF, *, END=160) IUD(4)
ELSE IF (COMM .EQ. 'EBY') THEN
READ (BUFF, *, END=160) IUD(5)
ELSE IF (COMM .EQ. 'EBZ') THEN
READ (BUFF, *, END=160) IUD(6)
ELSE IF (COMM .EQ. 'END') THEN
  ICOMM = 0
  ASSIGN 150 TO EXIT
  GO TO 180
150 CONTINUE
RETURN
ELSE IF (COMM .EQ. 'COORD') THEN
    CALL LOCORD (IDIM, H, ICOMM, LDEV11, IOUT, XAX, YAX, ZAX)
    XAXR(1) = XAX(1)
    YAXR(1) = YAX(1)
    ZAXR(1) = ZAX(1)
    XAXR(2) = XAX(2)
    YAXR(2) = YAX(2)
    ZAXR(2) = ZAX(2)
    XAXR(3) = XAX(3)
    YAXR(3) = YAX(3)
    ZAXR(3) = ZAX(3)

C
   IORD = 1
   IORD = 1
   IF (ICOMM .EQ. 1) THEN
      GO TO 120
   ELSE IF (ICOMM .EQ. 2) THEN
      ASSIGN 210 TO NEXT
      GO TO 180
   ELSE
      GO TO 100
ENDIF
ELSE IF (COMM .EQ. 'DCORD') THEN
    CALL LOCORD (IDIM, H, ICOMM, LDEV11, IOUT, XAX, YAX, ZAX)
    IORD = 1
    IF (ICOMM .EQ. 1) THEN
      GO TO 120
    ELSE IF (ICOMM .EQ. 2) THEN
      ASSIGN 210 TO NEXT
      GO TO 180
    ELSE
      GO TO 100
ENDIF
ELSE IF (COMM .EQ. 'RCORD') THEN
    CALL LOCORD (IDIM, H, ICOMM, LDEV11, IOUT, XAXR, YAXR, ZAXR)
    IORD = 1
    IF (ICOMM .EQ. 1) THEN
      GO TO 120
    ELSE IF (ICOMM .EQ. 2) THEN
      ASSIGN 210 TO NEXT
      GO TO 180
    ELSE
      GO TO 100
ENDIF
ELSE IF (COMM .EQ. 'COMB') THEN
    GO TO 100
ELSE
   ICOMM = 1
   ASSIGN 210 TO NEXT
   GO TO 180
ENDIF
C
GO TO 170
160 CONTINUE
   CALL ERRORS (3, 1, 'IDISP')
170 CONTINUE
   IF (H .NE. 0) THEN
      GO TO 110
   ELSE
      GO TO 100

EHDIF

C 180 CONTINUE
DO 200 K1 = ISTART, IEED, INTR
C Identify the node as a support
C
C ISPBK(K1) = IBSET(ISPB(K1),16)
ID1 = NWDF*(X1-1)
DO 190 IDIR = 1, NWDF
ID = ID1 + IDIR
IF (D(IDIR) .LT. 0.) U(ID) = D(IDIR)
IF (UD(IDIR) .LT. 0.) IDUD(ID) = IDUD(IDDIR)
IF (IDUD(IDDIR) .EQ. 0.) IDUD(ID) = 3
IF (IDO(IDDIR) .LT. 0) IDO(ID) = IDO(IDIR)
190 CONTINUE
C
C ---- ISPBK(k1) identifies whether any coordinate systems have been
C defined for the nodal displacements and rotations. It also
C stores information on the sign of the third direction cosine
C of each coordinate axis.
C
BIT 0 local coordinate system for translations
C
BIT 1 local coordinate system for rotations is defined
C
BIT 2 0; shell rotations in local shell coordinate system
C 1; shell rotations in global coordinate system.
C
BIT 3 sign bit for the third direction cosine of the
C local translation X-axis (0=+,1=-).
C
BIT 4 sign bit for the third direction cosine of the
C local translation Y-axis (0=+,1=-).
C
BIT 5 sign bit for the third direction cosine of the
C local rotation X-axis (0=+,1=-).
C
BIT 6 sign bit for the third direction cosine of the
C local rotation Y-axis (0=+,1=-).
C
BIT 7 sign bit for the third direction cosine of the
C shell rotation Z-axis (0=+,1=-).
C
IF (ICORD .EQ. 1) THEN
ISP = BTSET(ISPB(K1),0)
IF (ISP) THEN
CALL ERRORS (4, 0, 'IODEP')
ELSE
ISPBK(K1) = ISPBK(K1) + 1
ENDIF
XAXIS(1,K1) = XAX(1)
YAXIS(1,K1) = YAX(1)
XAXIS(2,K1) = XAX(2)
YAXIS(2,K1) = YAX(2)
C
C ---- set the sign bits for the third direction cosines of the
C of the local translation x and y axes.
C
IF (XAX(3) .LT. 0.) ISPBK(K1) = ISPBK(K1) + 8
IF (YAX(3) .LT. 0.) ISPBK(K1) = ISPBK(K1) + 16
ENDIF
C
IF (ICORD .EQ. 1) THEN
ISP = BTSET(ISPB(K1),1)
IF (ISP) THEN
CALL ERRORS (4, 0, 'IODEP')
ELSE
ISPBK(K1) = ISPBK(K1) + 2
ENDIF
XAXIS(3,K1) = XAX(3)
YAXIS(3,K1) = YAX(3)
XAXIS(4,K1) = XAX(4)
YAXIS(4,K1) = YAX(4)

C ---- set the sign bits for the third direction cosines of the
C of the local rotation x and y axes.
C
IF (XAX(3) .LT. 0.) ISPBCK1) = ISPBCK1) + 32
IF (YAX(3) .LT. 0.) ISPBCK1) = ISPBCK1) + 64

ENDIF

200 CONTINUE
GO TO NEXT

210 CONTINUE
RETURN

C
=================================================================
IMPLICIT REAL*4 (A-H,O-Z)

PROGRAM ICORD:
ICORD reads and process the local coordinate system
definitions.

ENTRY:
IDIM = physical dimension of the problem (i.e. 2D or 3D)
B = 0; buffer contains no additional commands
1; Buffer contains at least one additional command
LDEV11 = unit number for the input file
IOUT = unit number for the output file

RETURN:
IN = 0; buffer contains no additional commands
1; Buffer contains at least one additional command
ICOMM = identifies the action to be taken when control is
returned to the calling routine .
=0; read the next input line
=1; The current command was not resolved by IDIS.
Calling routine should process this command
=2; end of file is reached, do not try to read any
more lines.
XAX(i) = single precision direction cosines of the X-axis
YAX(i) = single precision direction cosines of the Y-axis
ZAX(i) = single precision direction cosines of the Z-axis

=================================================================

IMPLICIT REAL*8 (A-H,O-Z)

C...SWITCHES: RESUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C COMPRO ERRORS UNIV DIRVEC CROSS DOTPRO
C***********************************************************************

CHARACTER*80 BUFFER,BUFF
CHARACTER*6 COMM
REAL*4 XYZ,PERM,YAX,ZAX
COMM/INPUT3/XYZ(3,10000)
COMM/COMP2/COMM,BUFFER,BUFF
DIMENSION XZ(3),Y1(3),Z1(3),XY(3),YZ(3)
DIMENSION VECT(3,6),PERM(3,3),ISTART(6),IEND(6),ID(6)
DIMENSION COORD1(3,6),COORD2(3,6),YAX(3),ZAX(3)
EQUIVALENCE (VECT(1,1),Y1(1)),(VECT(1,2),Z1(1)),(VECT(1,3),XZ(1))
EQUIVALENCE (VECT(1,4),XY(1)),(VECT(1,5),YZ(1)),(VECT(1,6),ZAX(1))
DATA PERM/0.,0.,1.,0.,0.,1./
ICOMM = 0
IDS = 0
IVEC = 0
CST = 3.141592653589793D0/180.D0
DO 100 K2 = 1, 6
    ID(K2) = 0
    ISTART(K2) = 0
    IEND(K2) = 0
    VECT(1,K2) = 0.DO
    VECT(2,K2) = 0.DO
    VECT(3,K2) = 0.DO
100 CONTINUE
C  IVEC = 0; when vectors are defined by components
C GE 0; when vectors are defined by FROM n TO m option
C IF (N .NE. 0) GO TO 120
110 CONTINUE
READ (LDIEV11, ' (A 80)>, END=130) BUFFER
120 CONTINUE
CALL COMPRO (N, HVAR)
C
IF (IVEC .GT. 0) THEN
    IF (COMM .EQ. 'FROM') THEN
        IF (HVAR .EQ. 1) THEN
            READ (BUFF, *) ISTART(IVEC)
        ELSE IF (HVAR .EQ. 3 .OR. HVAR .EQ. IDIM) THEN
            READ (BUFF, *) (COORD1(K1,IVEC), K1 = 1, HVAR)
            ISTART(IVEC) = -1
        ELSE IF (HVAR .EQ. 0) THEN
            CALL ERRORS (3, 1, 'COORD')
        ENDIF
    ELSE IF (COMM .EQ. 'TO') THEN
        IF (HVAR .EQ. 1) THEN
            READ (BUFF, *) IEND(IVEC)
        ELSE IF (HVAR .EQ. 3 .OR. HVAR .EQ. IDIM) THEN
            READ (BUFF, *) (COORD2(K1,IVEC), K1 = 1, HVAR)
            IEND(IVEC) = -1
        ELSE IF (HVAR .EQ. 0) THEN
            CALL ERRORS (3, 1, 'COORD')
        ENDIF
    ENDIF
    IVEC = 0
ENDIF
ELSE
    IVEC = 0
    CALL ERRORS (7, 1, 'IOCORD')
ENDIF
ELSE IF (COMM.EQ. 'RECTAN' .OR. COMM.EQ. 'CARTES') THEN
    ELSE IF (COMM.EQ. 'CYLIND') THEN
    ELSE IF (COMM.EQ. 'SPHERI') THEN
    ELSE IF (COMM.EQ. 'XVECTO' .OR. COMM.EQ. 'XAXIS') THEN
      IVEC = 0
      ID(1) = 1
      IF (BVAR.EQ.IDIM .OR. BVAR.EQ.3) THEN
        READ (BUFF, *) (X1(K1), K1 = 1, IDIM)
      ELSE IF (BVAR.EQ.0) THEN
        IVEC = 1
      ELSE
        CALL ERRORS (7, 1, 'IOCORD')
      ENDIF
    ELSE IF (COMM.EQ. 'YVECTO' .OR. COMM.EQ. 'YAXIS') THEN
      IVEC = 0
      ID(2) = 1
      IF (BVAR.EQ.IDIM .OR. BVAR.EQ.3) THEN
        READ (BUFF, *) (Y1(K1), K1 = 1, IDIM)
      ELSE IF (BVAR.EQ.0) THEN
        IVEC = 2
      ELSE
        CALL ERRORS (7, 1, 'IOCORD')
      ENDIF
    ELSE IF (COMM.EQ. 'ZVECTO' .OR. COMM.EQ. 'ZAXIS') THEN
      IVEC = 0
      ID(3) = 1
      IF (BVAR.EQ.IDIM .OR. BVAR.EQ.3) THEN
        READ (BUFF, *) (Z1(K1), K1 = 1, IDIM)
      ELSE IF (BVAR.EQ.0) THEN
        IVEC = 3
      ELSE
        CALL ERRORS (7, 1, 'IOCORD')
      ENDIF
    ELSE IF (COMM.EQ. 'XYVECT') THEN
      IVEC = 0
      ID(4) = 1
      IF (BVAR.EQ.IDIM .OR. BVAR.EQ.3) THEN
        READ (BUFF, *) (XY(K1), K1 = 1, IDIM)
      ELSE IF (BVAR.EQ.0) THEN
        IVEC = 4
      ELSE
        CALL ERRORS (7, 1, 'IOCORD')
      ENDIF
    ELSE IF (COMM.EQ. 'XZVECT') THEN
      IVEC = 0
      ID(6) = 1
      IF (BVAR.EQ.IDIM .OR. BVAR.EQ.3) THEN
        READ (BUFF, *) (XZ(K1), K1 = 1, IDIM)
      ELSE IF (BVAR.EQ.0) THEN
        IVEC = 6
      ELSE
        CALL ERRORS (7, 1, 'IOCORD')
      ENDIF
    ELSE IF (COMM.EQ. 'YZVECT') THEN
      IVEC = 0
      ID(5) = 1
      IF (BVAR.EQ.IDIM .OR. BVAR.EQ.3) THEN
        READ (BUFF, *) (YZ(K1), K1 = 1, IDIM)
      ELSE IF (BVAR.EQ.0) THEN
        IVEC = 5
      ELSE
        CALL ERRORS (7, 1, 'IOCORD')
      ENDIF
    ELSE
    END
IVEC = 5
ELSE
CALL ERRORS (7, 1, 'IDCORD')
ENDIF
ELSE IF (COMM .EQ. 'COMMEH') THEN
GO TO 110
ELSE
ICOMM = 1
GO TO 140
ENDIF
C
IF (S .NE. 0) THEN
GO TO 120
ELSE
GO TO 110
ENDIF
C
130 CONTINUE
ICOMM = 2
140 CONTINUE
ID1 = 0
ID2 = 0
DO 150 K1 = 1, 6
IF (ID1 .EQ. 0) THEN
IF (ID(K1) .EQ. 1) ID1 = K1
ELSE IF (ID2 .EQ. 0) THEN
IF (ID(K1) .EQ. 1) ID2 = K1
ELSE
GO TO 160
ENDIF
150 CONTINUE
C
160 CONTINUE
IDD = ID1
DO 170 K1 = 1, 2
IF (ISTART(IDD) .NE. 0) THEN
IF (ISTART(IDD) .GT. 0) THEN
COORD1(1,IDD) = XYZ(1,ISTART(IDD))
COORD1(2,IDD) = XYZ(2,ISTART(IDD))
COORD1(3,IDD) = XYZ(3,ISTART(IDD))
ENDIF
C
IF (IEHD(IDD) .GT. 0) THEN
COORD2(1,IDD) = XYZ(1,IEHD(IDD))
COORD2(2,IDD) = XYZ(2,IEHD(IDD))
COORD2(3,IDD) = XYZ(3,IEHD(IDD))
ENDIF
VECT(1,IDD) = COORD2(1,IDD) - COORD1(1,IDD)
VECT(2,IDD) = COORD2(2,IDD) - COORD1(2,IDD)
VECT(3,IDD) = COORD2(3,IDD) - COORD1(3,IDD)
ENDIF
170 CONTINUE
C
IF (ID1 .LT. 3 .AND. ID2 .EQ. 0) THEN
CALL DIRVEC (VECT(1,1), VECT(1,2), VECT(1,3))
ELSE IF (ID1 .EQ. 3) THEN
CALL DIRVEC (VECT(1,2), VECT(1,1), VECT(1,3))
ELSE IF (ID1 .EQ. 2) THEN
CALL DIRVEC (VECT(1,3), VECT(1,1), VECT(1,2))
ELSE IF (ID1 .EQ. 3) THEN
CALL DIRVEC (VECT(1,1), VECT(1,2), VECT(1,3))
ELSE
    CALL ERRORS (6, 1, 'IDCORD')
ENDIF
ELSE IF (ID1 .GT. 3.) THEN
    CALL ERRORS (6, 1, 'IDCORD')
ELSE IF (ID1 .EQ. ID2) THEN
    CALL ERRORS (6, 1, 'IDCORD')
ELSE IF (ID1 .LT. 3) THEN
    IF (ID2 .LT. 3) THEN
        GO TO 180
    ELSE IF (ID1 .EQ. 1) THEN
        IF (ID2 .EQ. 6) THEN
            Y1(1) = YZ(1)
            Y1(2) = YZ(2)
            Y1(3) = YZ(3)
            ID2 = 2
        ELSE IF (ID2 .EQ. 4) THEN
            CALL CROSS (X1, XY, Z1)
            ID2 = 3
        ELSE IF (ID2 .EQ. 6) THEN
            CALL CROSS (IZ, X1, Y1)
            ID2 = 2
        ENDIF
    ELSE IF (ID2 .EQ. 4) THEN
        IF (ID1 .EQ. 6) THEN
            X1(1) = XZ(1)
            X1(2) = XZ(2)
            X1(3) = XZ(3)
            ID2 = 1
        ELSE IF (ID1 .EQ. 4) THEN
            CALL CROSS (XY, Y1, Z1)
            ID2 = 3
        ELSE IF (ID1 .EQ. 6) THEN
            CALL CROSS (Y1, YZ, X1)
            ID2 = 1
        ENDIF
    ENDIF
ELSE IF (ID1 .EQ. 2) THEN
    IF (ID2 .EQ. 6) THEN
        X1(1) = XZ(1)
        X1(2) = XZ(2)
        X1(3) = XZ(3)
        ID2 = 1
    ELSE IF (ID2 .EQ. 4) THEN
        CALL CROSS (YZ, Z1, X1)
        ID2 = 1
    ELSE IF (ID2 .EQ. 6) THEN
        CALL CROSS (Z1, XZ, Y1)
        ID2 = 2
    ENDIF
ENDIF
ENDIF
C 180 CONTINUE
CALL UNITV (VECT(1,ID1),3)
CALL UNITV (VECT(1,ID2),3)
DOT = DOTPRO(VECT(1,ID1),VECT(1,ID2),3)
IF (DABS(DOT) .GT. 1.0E-4) CALL ERRORS (6, 1, 'IDCORD')
C DO 190 K1 = 1, 3
    IF (K1.NE.ID1 .AND. K1.NE.ID2) ID3 = K1
190 CONTINUE
C CALL CROSS (VECT(1,ID1),VECT(1,ID2),VECT(1,ID3))
COHST = PERM(ID1,ID2)
VECT(1,ID3) = VECT(1,ID3)+COHST
VECT(2,ID3) = VECT(2,ID3)+COHST
VECT(3,ID3) = VECT(3,ID3)+COHST
C
XAX(1) = X1(1)
YAX(1) = Y1(1)
ZAX(1) = Z1(1)
XAX(2) = X1(2)
YAX(2) = Y1(2)
ZAX(2) = Z1(2)
XAX(3) = X1(3)
YAX(3) = Y1(3)
ZAX(3) = Z1(3)
C
ENDIF
C
200 CONTINUE
RETURN
END
C
================================================================
C
PROGRAM:
C
ILOAD reads and stores the structural loads in either the
C
global or the local coordinate system.
C
ENTRY:
C
IDIM = physical dimension of the problem (i.e. 2D or 3D)
NDF = number of nodal degrees of freedom
N = 0; buffer contains no additional commands
1; Buffer contains at least one additional command
LDEV11 = unit number for the input file
IGUT = unit number for the output file
C
RETURN:
C
N = 0; buffer contains no additional commands
1; Buffer contains at least one additional command
ICOMM = identifies the action to be taken when control is
returned to the calling routine.
0; read the next input line
1; The current command was not resolved by IODIS.
Calling routine should process this command
2; end of file is reached, do not try to read any
more lines.
C
================================================================
C
IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: REFUMB=100:10,FORMAT=900:10
C...SWITCHES:
C
C
SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

COMPRO  ICOND  ERRORS

***********************

CHARACTER*80 BUFFER,BUFF
CHARACTER*6 COMM
REAL*4 XAX,YAX,ZAX,XAXR,YAXR,ZAXR
COMM/COMP2/COMM,BUFFER,BUFF
COMM/SAFE2/R(60000),IDOF(60000),JDIAG(60000)
COMM/SAFE4/IDOF(60000),JDOF(60000)
DIMENSION P(6),XAX(3),YAX(3),ZAX(3),XAXR(3),YAXR(3),ZAXR(3),IRD(6)

ITIME = 0
ICORD = 0
ICOMM = 0
ISTART = 0
IEND = 0
INTR = 0

IF (H .GE. 0) GO TO 110
100 CONTINUE
READ (LDEV11, '*(A80)', END=190) BUFFER
110 CONTINUE
CALL COMPRO (H, EVAR)

120 CONTINUE
IF (COMM.EQ.'HODE' .OR. COMM.EQ.'HODES') THEN
IF (ITIME .EQ. 1) THEN
ASSIGN 130 TO NEXT
GO TO 160
ENDIF
130 CONTINUE
READ (BUFF, *, END=200) ISTART
ITIME = 1
IEND = ISTART
INTR = 1

---- ICOND is the flag for the translation coordinate definition.
COND = 0; a local coordinate system is not defined
C = 1; a local coordinate system is defined
C ---- ICOND is the flag for the rotational coordinate definition.
COND = 0; a local coordinate system is not defined
C = 1; a local coordinate system is defined

ICORD = 0
ICOMM = 0
DO 140 K1 = 1, HHDF
IRD(K1) = 0.
P(K1) = 0.
140 CONTINUE
ELSE IF (COMM.EQ.'TO') THEN
READ (BUFF, *, END=200) END
ELSE IF (COMM.EQ.'BY') THEN
READ (BUFF, *, END=200) INTR
ELSE IF (COMM.EQ.'FX') THEN
READ (BUFF, *, END=200) P(1)
ELSE IF (COMM.EQ.'FY') THEN
READ (BUFF, *, END=200) P(2)
ELSE IF (COMM.EQ.'FZ') THEN
READ (BUFF, *, END=200) P(3)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ELSE IF (COMM .EQ. 'MX') THEN
   READ (BUFF, *, END=200) P(4)
ELSE IF (COMM .EQ. 'MY') THEN
   READ (BUFF, *, END=200) P(5)
ELSE IF (COMM .EQ. 'MZ') THEN
   READ (BUFF, *, END=200) P(6)
ELSE IF (COMM .EQ. 'SFY') THEN
   READ (BUFF, *, END=200) P(8)
ELSE IF (COMM .EQ. 'SFZ') THEN
   READ (BUFF, *, END=200) IRD(1)
ELSE IF (COMM .EQ. 'HMY') THEN
   READ (BUFF, *, END=200) IRD(2)
ELSE IF (COMM .EQ. 'HMY') THEN
   READ (BUFF, *, END=200) IRD(3)
ELSE IF (COMM .EQ. 'EMY') THEN
   READ (BUFF, *, END=200) IRD(4)
ELSE IF (COMM .EQ. 'ERM') THEN
   READ (BUFF, *, END=200) IRD(5)
ELSE IF (COMM .EQ. 'ERZ') THEN
   READ (BUFF, *, END=200) IRD(6)
ELSE IF (COMM .EQ. 'COORDI') THEN
   CALL LOCORD (IDIM, E, ICOMM, LDEV11, IOUT, XAX, YAX, ZAX)
   XAXR(1) = XAX(1)
   YAXR(1) = YAX(1)
   ZAXR(1) = ZAX(1)
   XAXR(2) = XAX(2)
   YAXR(2) = YAX(2)
   ZAXR(2) = ZAX(2)
   XAXR(3) = XAX(3)
   YAXR(3) = YAX(3)
   ZAXR(3) = ZAX(3)
C
   IORD = 1
   IORD = 1
IF (ICOMM .EQ. 1) THEN
   GO TO 120
ELSE IF (ICOMM .EQ. 2) THEN
   ASSIGN 190 TO EXIT
   GO TO 160
ELSE
   GO TO 100
ENDIF
ELSE IF (COMM .EQ. 'DCORD') THEN
   CALL LOCORD (IDIM, E, ICOMM, LDEV11, IOUT, XAX, YAX, ZAX)
   IORD = 1
IF (ICOMM .EQ. 1) THEN
   GO TO 120
ELSE IF (ICOMM .EQ. 2) THEN
   ASSIGN 190 TO EXIT
   GO TO 160
ELSE
   GO TO 100
ENDIF
ELSE IF (COMM .EQ. 'ECORD') THEN
   CALL LOCORD (IDIM, E, ICOMM, LDEV11, IOUT, ZAER, YAXR, ZAXR)
   IORD = 1
IF (ICOMM .EQ. 1) THEN
   GO TO 120
ELSE IF (ICOMM .EQ. 2) THEN
   ASSIGN 190 TO EXIT
   GO TO 160
ELSE
   GO TO 100
ENDIF
ELSE IF (COMM .EQ. 'EEND') THEN
ICOMM = 0
ASSIGN 160 TO NEXT
GO TO 160
160 CONTINUE
RETURN
ELSE IF (COMM .EQ. 'COMM') THES
GO TO 100
ELSE
ICOMM = 1
ASSIGN 160 TO NEXT
GO TO 160
ENDIF
C
IF (H .NE. 0) THEN
GO TO 110
ELSE
GO TO 100
ENDIF
C
160 CONTINUE
IF (ICOORD .EQ. 1) THEN
CST1 = IAX(1)*P(1) + YAX(1)*P(2) + ZAX(1)*P(3)
CST2 = IAX(2)*P(1) + YAX(2)*P(2) + ZAX(2)*P(3)
CST3 = IAX(3)*P(1) + YAX(3)*P(2) + ZAX(3)*P(3)
P(1) = CST1
P(2) = CST2
P(3) = CST3
ENDIF
C
IF (ICOORD .EQ. 1) THEN
CST1 = IAXR(1)*P(4) + YAXR(1)*P(5) + ZAXR(1)*P(6)
CST2 = IAXR(2)*P(4) + YAXR(2)*P(5) + ZAXR(2)*P(6)
CST3 = IAXR(3)*P(4) + YAXR(3)*P(5) + ZAXR(3)*P(6)
P(4) = CST1
P(5) = CST2
P(6) = CST3
ENDIF
C
DO 180 K1 = ISTART, IEND, ISTR
ID1 = EDF*(K1-1)
DO 170 IDIR = 1, HNDF
ID = ID1 + IDIR
IF (IRD(IDIR) .NE. 0.) IRDOF(ID) = IRD(IDIR)
IF (IRD(IDIR) .EQ. 0.) IRDOF(ID) = 3
IF (P(IDIR) .NE. 0.) R(ID) = P(IDIR)
170 CONTINUE
180 CONTINUE
C
GO TO NEXT
C
190 CONTINUE
ICOMM = 2
RETURN
200 CONTINUE
CALL ERRORS (3, 1, 'ILOAD')
STOP
END
C
C ==ELEMEH==
C
SUBROUTINE ELEMEH(H,ICOMM,LDEV11,LOUT,BSELEM)
PROGRAM ELEHES reads and stores physical properties of the elements

ENTRY:

DIM = physical dimension of the problem (i.e. 2D or 3D)

N = 0; buffer contains no additional commands
1; Buffer contains at least one additional command

LEV = unit number for the input file

OUT = unit number for the output file

RETURN:

N = 0; buffer contains no additional commands
1; Buffer contains at least one additional command

COMM = identifies the action to be taken when control is

=0; read the next input line

=1; The current command was not resolved by IO DIS.

=2; end of file is reached, do not try to read any

more lines.

HELEM = temporary storage for number of nodes in the elements.

*********************************************************************

IMPLICIT REAL*8 (A-H,O-Z)

SWITCHES: RENUHB=100:10, F0RHAT=900:10

SWITCHES:

SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

COMPRO ERRORS

*********************************************************************

CHARACTER*80 BUFFER, BUFF

CHARACTER*6 COMM

c changes 7/18

c === REAL*4 ETICK

c changes 7/18

REAL*4 THICK, T1, T2, T3, T4

COMMON/INPUTA/IHFEOEL(6,8000)

COMMON/COMP2/COMM, BUFFER, BUFF

DIMENSION HELEM(*

c changes 7/18

EQUIVALENCE (T1, IT1), (T2, IT2), (T3, IT3), (T4, IT4)

c changes 7/18

C ---- read and generate element information

IFLAG = 0

COMM = 0

LOC = 0

ITF = 0
READ (BUFF, *, END=160) ISTART
C
C changes 7/18
T1 = 0.0
T2 = 0.0
T3 = 0.0
T4 = 0.0
C changes 7/18
IEHD = ISTART
IHTR = 1
MAT = 0
HIPXI = 0
HIPETA = 0
HIPSI = 0
INTCOD = 0
THICK = 0.
ITYPE = 0.
IF (N .NE. 0) GO TO 110
100 CONTINUE
READ (LDEV11, '"(A80)"', END=120) BUFFER
110 CONTINUE
CALL COMPRO (N, NVAR)
C
IF (COMM .EQ. 'TO') THEN
READ (BUFF, *, END=160) IEND
ELSE IF (COMM .EQ. 'BY') THEN
READ (BUFF, *, END=160) IHTR
ELSE IF (COMM .EQ. 'HIPXI') THEN
READ (BUFF, *, END=160) HIPXI
ELSE IF (COMM .EQ. 'HIPETA') THEN
READ (BUFF, *, END=160) HIPETA
ELSE IF (COMM .EQ. 'HIPZET') THEN
READ (BUFF, *, END=160) HIPSI
ELSE IF (COMM .EQ. 'THICK') THEN
READ (BUFF, *, END=160) THICK
ITF = 30
T1 = THICK
T2 = THICK
T3 = THICK
T4 = THICK
ELSE IF (COMM .EQ. 'T1') THEN
READ (BUFF, *, END=160) T1
ITF = ITF + 2
ELSE IF (COMM .EQ. 'T2') THEN
READ (BUFF, *, END=160) T2
ITF = ITF + 4
ELSE IF (COMM .EQ. 'T3') THEN
READ (BUFF, *, END=160) T3
ITF = ITF + 8
ELSE IF (COMM .EQ. 'T4') THEN
READ (BUFF, *, END=160) T4
ITF = ITF + 16
ELSE IF (COMM .EQ. 'MATERI') OR COMM .EQ. 'MAT') THEN
READ (BUFF, *, END=160) MAT
IF (MAT .GT. 7) CALL ERRORS (13, 1, 'ELEMENT')
ELSE IF (COMM .EQ. 'TYPE') THEN
READ (BUFF, *, END=160) ITYPE
ELSE IF (COMM .EQ. 'COMMEH') THEN
GO TO 100
ELSE
ICOMM = 1
GO TO 130
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
422

ENDIF

C IF (W .NE. 0) THEN
GO TO 110
ELSE
GO TO 100
ENDIF

C

C ---- Condensed data storage in array INFSEL.
C
C ---- Bit storage organization for INFSEL(1,enum)
C
C Bit Range Max. Value Description
C
C 0 - 2 7 material number
C 3 - 8 63 number of nodes in the element
C 9 - 17 511 element type number
C 18 - 20 7 analysis type flag
C 21 - 23 7 integration points in XI direc.
C 24 - 26 7 integration points in ETA direc.
C 27 - 30 7 integration points in ZETA dir.

C ---- Bit storage organization for INFSEL(2,enum)
C
C Bit Range Max. Value Description
C
C 0 - 7 255 integration code
C 8 - 14 128 number of layers
C 15 - 16 3 flag for layering.
C 17 - 22 63 number of lines connecting nodes
C 23 - 30 255 starting position of line
C connectivity in arrays ISTART & IEND

C C ---- Table of internal element type numbers
C
C Type No. No. of nodes Description
C
C 204 4 linear isoparametric quadrilateral
C 208 8 quadratic isoparametric quadrilateral
C 209 9 Lagrangian quadratic isop. quad.
C 219 9 quadratic to cubic isop. quad.
C 222 4 linear isoparametric brick or
C 308 8 quadratic shell
C 309 9 Lagrangian quadratic shell
C 320 20 Lagrangian quadratic brick

120 CONTINUE
ICOMM = 2
130 CONTINUE
INSEL = 0
IF (ITYPE .NE. 0) THEN
ID = ITYPE/1000
ID1 = ITYPE - ID*1000
IFLAG = ID1/100
INSEL = ID1 - IFLAG*100
ITYPE = ID*100 + INSEL
IF (ITYPE .LT. 300) THEN
IF (INSEL .EQ. 4) THEN

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
LOCATE = 1
LINES = 4

ELSE IF (HHEL .EQ. 6) THEN
LOCATE = 5
LINES = 6

ELSE IF (ITYPE.EQ.208 .OR. ITYPE.EQ.209) THEN
LOCATE = 10
LINES = 8

ELSE IF (ITYPE .EQ. 219) THEN
HHEL = 9
LOCATE = 54
LINES = 9

ELSE
CALL ERRORS (12, 1, 'ELEMEH')
ENDIF

ELSE IF (ITYPE .GT. 300) THEN
IF (IFLAG .EQ. 0) THEN
IF (HHEL .EQ. 8) THEN
LOCATE = 18
LINES = 12
ELSE IF (HHEL .EQ. 20) THEN
LOCATE = 30
LINES = 24
ELSE
CALL ERRORS (12, 1, 'ELEMEH')
ENDIF
ELSE IF (ITYPE .EQ. 308 .OR. ITYPE.EQ.309) THEN
LOCATE = 10
LINES = 8
ELSE IF (ITYPE .EQ. 304) THEN
LOCATE = 1
LINES = 4
ELSE
CALL ERRORS (12, 1, 'ELEMEH')
ENDIF
ELSE
CALL ERRORS (12, 1, 'ELEMEH')
ENDIF
ELSE
CALL ERRORS (12, 1, 'ÈLEMEH')
ENDIF
ENDIF

C ABD operations are used to clear the bits prior to storage
C of the appropriate values.

DO 140 K = ISTART, IEND, IBTR
IF (ITYPE .NE. 0) THEN
IFN1 = IABD(INFOEL(1,K),2146386803)
INFOEL(1,K) = INF1 + HHEL*B8 + ITYPE*512 + IFLAG*262144
BBELEH(K) = BHEL
IBF0EL(2,K) = IAHD(INFOEL(2,K),2129226111)
INFOEL(2,K) = INF2 + 131072*LIBES
IFF2 = IABD(INFOEL(2,K),8388607)
INFOEL(2,K) = INF2 + 8388608*LOCATE
ENDIF
IF (MAT .NE. 0) THEN
INF1 = IABD(INFOEL(1,K),2147483640)
INFOEL(1,K) = INF1 + MAT
ENDIF
IF (HIPXI .NE. 0) THEN
INFOEL(2,K) = IABD(INFOEL(2,K),2147483392)
IBFl = IAND(IHFDEL(1,K),2132803683)
INFDEL(1,K) = IBFl + 2097152*HIPZI

EHDIF
IF (HIPETA .GE. 0) THEN
INFDEL(2,K) = IAND(INFDEL(2,K),2147483392)
INFDEL(1,K) = INFDEL(1,K) + 16777216*HIPETA
ENDIF

IF (HIPSI .GE. 0) THEN
INFDEL(2,K) = IAND(INFDEL(2,K),2147483392)
INFDEL(1,K) = INFDEL(1,K) + 134217728*HIPSI
ENDIF

IF (IHTCOD .GE. 0) THEN
INFDEL(2,K) = IAND(INFDEL(2,K),2147483392)
INFDEL(1,K) = INFDEL(1,K) + 134217728*IHTCOD
ENDIF

IF (BTEST(ITF,1)) INFDEL(3,K) = IT1
IF (BTEST(ITF,2)) INFDEL(4,K) = IT2
IF (BTEST(ITF,3)) INFDEL(5,K) = IT3
IF (BTEST(ITF,4)) INFDEL(6,K) = IT4

C changes 7/18
140 CONTINUE
C
150 CONTINUE
RETURN
160 CONTINUE
CALL ERRORS (3, 1, 'ELEMEHO')
END

C -------------------------- I O F R E E --------------------------
C
C INCLUDE (PROCESS)
SUBROUTINE IOFREE(H,ICOMM,LDEV11,IOUT)
C
C PROGRAM:
C IOFREE reads the freebody definitions
C
C ENTRY:
C IDIM = physical dimension of the problem (i.e. 2D or 3D)
C B = 0; buffer contains no additional commands
C 1; Buffer contains at least one additional command
C LDEV11 = unit number for the input file
C IOUT = unit number for the output file
C
C RETURN:
C B = 0; buffer contains no additional commands
C 1; Buffer contains at least one additional command
C ICOMM = identifies the action to be taken when control is
C returned to the calling routine .
C 0; read the next input line
C =1; The current command was not resolved by IDDIS.
C =2; end of file is reached, do not try to read any
C more lines.
IMPLICIT REAL*8 (A-H, O-Z)
C...SWITCHES: RENUM=100:10,FORMAT=900:10
C...SWITCHES:
C******************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C******************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C******************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C******************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C******************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C******************************************************************************

C ------ read and generate freebody information
C
ISTART = 0
IEND = 0
INTR = 0
HVAR = 0
ITIME = 0
ICOMM = 0
READ (BUFF, *, EHD=190) HUH
HFREE = HAXO(HFREE,HUM)
C
IF (H .GE. 0) GO TO 110
100 CONTINUE
READ (LDEV1, '(A80)', EHD=150) BUFFER
110 CONTINUE
CALL COMPRO (H, HVAR)
C
IF (COMM.EQ. 'ELEM' .OR. COMM.EQ. 'ELEMEH') THEN
IF (HVAR .EQ. 1) THEN
IF (ITIME .EQ. 1) THEN
ASSIGN 120 TO HEXT
GO TO 160
ENDIF
ITIME = 1
C
120 CONTINUE
READ (BUFF, *, EHD=190) ISTART
INTR = 1
IEND = ISTART
ELSE IF (HVAR .GT. 1) THEN
IF (ITIME .EQ. 1) THEN
ASSIGN 130 TO HEXT
GO TO 160
ENDIF
ITIME = 0
130 CONTINUE
READ (BUFF, *, EHD=190) (IELEM(K1), K1 = 1, HVAR)
DO 140 K1 = 1, HVAR
   IF(BODY(IELEM(K1))) = IBSIT(IFBODY(IELEM(K1)),NUM)
140 CONTINUE
IF (N .NE. 0) THEN
  GO TO 110
ELSE
  GO TO 100
ENDIF
ELSE IF (COMM .EQ. 'TO') THEN
  READ (BUFF, *, END=190) lEDD
  IF (ISTART .EQ. 0) CALL ERRORS (19, 1, 'lOFREE')
ELSE IF (COMM .EQ. 'BY') THEN
  READ (BUFF, *, END=190) IBTR
ELSE IF (COMM .EQ. 'CDMMEN') THEN
  GO TO 100
ELSE
  ICOMM = 1
  ASSIGN 180 TO NEXT
  GO TO 160
ENDIF
IF (N .NE. 0) THEN
  GO TO 110
ELSE
  GO TO 100
ENDIF
C
C ----- Condensed data storage in array IFBODY.
C
C ----- Bit storage organization for IFBODY
C
Bit Range | Entity | Description
----------|--------|----------------
1-15      | element| 0; does not belong to freebody
          |        | 1; belongs to freebody
16-30     | nodes  | 0; does not belong to freebody
          |        | 1; belongs to freebody
C
For elements freebody number is the bit number.
For nodes freebody number is bit number minus 15.

150 CONTINUE
  ICOMM = 2
  ASSIGN 180 TO NEXT
160 CONTINUE
  DO 170 K1 = ISTART, IEND, IBTR
  IFBODY(K1) = IBSETdFBODY(K1),NUH)
  170 CONTINUE
  GO TO NEXT
C
180 CONTINUE
RETURN
190 CONTINUE
CALL ERRORS (3, 1, 'IOMATR')
END
PROGRAM:

IOMATR reads and stores the material properties.

ON ENTRY:

N = 0; buffer contains no additional commands
1; Buffer contains at least one additional command

LDEV11 = unit number for the input file
IOUT = unit number for the output file

ON RETURN:

N = 0; buffer contains no additional commands
1; Buffer contains at least one additional command

ICOMM = identifies the action to be taken when control is
returned to the calling routine.
0; read the next input line
1; the current command was not resolved by IOMATR.
2; Calling routine should process this command
3; end of file is reached, do not try to read any
more lines.

=================================================================

IMPLICIT REAL*8 (A-H,O-Z)

SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

COMPRO ERRORS

CHARACTER*80 BUFFER,BUFF
CHARACTER*6 COMM
REAL*8 HUX,HUY,HUZ
COMM/INPUT/MATTYPE(10)
COMM/INPUT/HUX(10),HUY(10),HUZ(10),EX(10),EY(10),EZ(10),
P1X(10),P1Y(10),P1Z(10),P2X(10),P2Y(10),P2Z(10)
COMM/COM2/COMM,BUFFER,BUFF

ICOMM = 0

READ (BUFF, *, END=140) MATSUM
IF ($ .NE. 0) GO TO 110
100 CONTINUE
READ (LDEV11, '(A80)') , END=120) BUFFER
110 CONTINUE
CALL COMPRO ($, EVAR)

IF (COMM .EQ. 'HUX') THEN
READ (BUFF, *, END=160) EUX(MATSUM)
HUX(MATSUM) = EUX(MATSUM)
ELSE IF (COMM .EQ. 'EY') THEN
READ (BUFF, *, END=160) EY(MATSUM)
EY(MATSUM) = EX(MATSUM)
EZ(MATNUM) = EX(MATNUM)
ELSE IF (COMM .EQ. 'WX') THEN
READ (BUFF, *, END=140) WGT1(MATNUM)
ELSE IF (COMM .EQ. 'WY') THEN
READ (BUFF, *, END=140) WGT2(MATNUM)
ELSE IF (COMM .EQ. 'WZ') THEN
READ (BUFF, *, END=140) WGT3(MATNUM)
ELSE IF (COMM .EQ. 'TYPE') THEN
READ (BUFF, *, END=140) MATYPE(MATNUM)
ELSE IF (COMM .EQ. 'YIELD') THEN
READ (BUFF, *, END=140) PIY(MATNUM)
ELSE IF (COMM .EQ. 'ISOT') THEN
READ (BUFF, *, END=140) PIY(MATNUM)
ELSE IF (COMM .EQ. 'KIBE') THEN
READ (BUFF, *, END=140) PIX(HATHUM)
ELSE IF (COMM .EQ. 'COMMEH') THEN
GO TO ICO
ELSE
ICOMM = 1
GO TO 130
ENDIF
C
IF (H .GE. 0) THEN
GO TO 110
ELSE
GO TO 100
ENDIF
C
120 CONTINUE
ICOMM = 2
130 CONTINUE
IF (MATYPE(MATNUM) .EQ. 0) THEN
WRITE (BUFFER, *) 'MATERIAL', HATHUM
CALL ERRORS (18, 1, 'lOMATR')
ENDIF
C
RETURN
140 CONTINUE
CALL ERRORS (3, 1, 'lOMATR')
STOP
END
C
C ============== GRAPH X ===============
C
C INCLUDE (PROCESS)
SUBROUTINE GRAPHX(K,ICOMM,LDEV11,LOUT)
C
C**SWITCHES: RESUMB=100:10,FORMAT=900:10
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C COMPO
C
C**************************************************************
CHARACTER*80 BUFFER,BUFF
CHARACTER*6 COMM
REAL*4 FMAG,DMAG
REAL*8 ABGLE,EIGHT
COMMON/GRAPHE/XL,XR,YB,YT,ZF,Z
COMMON/GRAPHE/FMAG,DMAG,EIGHT,ABGLE,DXLINE,ITHICK,ETLINES
COMMON/COMP2/COMM,BUFFER,BUFF

READ THE COMMAND LINE BUFFER

ICOMM = 0
IF ( I .NE. 0 ) GO TO 110
100 CONTINUE
READ (LDEV11, '(A80)', END=120) BUFFER
110 CONTINUE
CALL COMPRO (H, HVAR)
IF ( H .EQ. 0 ) THEN
ASSIGN 100 TO EXIT
ELSE
ASSIGN 110 TO EXIT
ENDIF

IF (COMM .EQ. 'FMAG') THEN
READ (BUFF, *, END=130) FMAG
ELSE IF (COMM .EQ. 'DMAG') THEN
READ (BUFF, *, END=130) DMAG
ELSE IF (COMM .EQ. 'WL') THEN
READ (BUFF, *, END=130) WL
ELSE IF (COMM .EQ. 'XR') THEN
READ (BUFF, *, END=130) XR
ELSE IF (COMM .EQ. 'YT') THEN
READ (BUFF, *, END=130) YT
ELSE IF (COMM .EQ. 'XB') THEN
READ (BUFF, *, END=130) XB
ELSE IF (COMM .EQ. 'THICK' .OR. COMM.EQ. 'THICKH') THEN
READ (BUFF, *, END=130) ITHICK
ELSE IF (COMM .EQ. 'IOOUT') THEN
WOLINE = 1
ELSE IF (COMM .EQ. 'EIGHT') THEN
READ (BUFF, *, END=130) EIGHT
ELSE IF (COMM .EQ. 'ANGLE') THEN
READ (BUFF, *, END=130) ANGLE
ELSE IF (COMM .EQ. 'COMHEH') THEN
GO TO 100
ELSE IF (COMM .EQ. 'END') THEN
RETURN
ELSE
ICOMM = 1
RETURN
ENDIF
GO TO EXIT
120 CONTINUE
ICOMM = 2
RETURN
130 CONTINUE
STOP

END

C ======================== I O U T ========================

SUBROUTINE IOOUT($,ICOMM,LDEV11)
IMPLICIT REAL*8(A-E,0-Z)
C...SWITCHES: REHUMB=100:10,FORMAT=900:10
C...SWITCHES:
C****************************************************************************
C C subroutines and functions called from this routine
C C********************COMPRO ERRORS *******************************************
C****************************************************************************
CHARACTER*80 BUFFER,BUFF
CHARACTER*6 COMM
COMMON/COMP2/COHM,BUFFER,BUFF
COMMON/INPUTG/IFLAG3.IOINTR.IFPLT
COMMON/OUTPT1/IOFLAG.IFOAED
COMMON/OUTPT2/IOEL(IOOO).IOnOD(lOOOO)

C Description of the output modules and the corresponding bit flag position in the IOFLAG integer variable

file bit description
O U T 1  1 output element stresses at the integration points
O U T 2  2 output element strains at the integration points
O U T 3  3 output element stresses at the element nodes
O U T 4  4 output element strains at the element nodes
O U T 5  5 output average stresses at the nodes
O U T 6  6 output average strains at the nodes
O U T 7  7 output displacements at the nodes
O U T 8  8 output nodal equilibrium forces at the nodes
O U T 9  9 output nodal equilibrium forces for freebodies
O U T 10 10 output reactions at the supports
O U T 11 11 output nodal coordinates
O U T 12 12 output element connectivity
O U T 20 20 output second Piola Kirchhoff stresses at the element nodes
O U T 21 21 output Cauchy stresses at the element nodes
O U T 22 22 output total element strains at the element nodes
O U T 23 23 output element elastic strains at the element node
O U T 24 24 output plastic element strains at the element node
O U T 25 25 output nodal displacements
O U T 26 26 output support reactions
O U T 27 27 output nodal equilibrium loads
READ (BUFF, *, END=180) IOINTR
100 CONTINUE
ICOMM = 0
ISTART = 0
IEND = 0
INTR = 1
IFG = 0
IF (H .NE. 0) GO TO 120
110 CONTINUE
READ (LDEVll, '(*), END=160) BUFFER
120 CONTINUE
CALL COMPRO (S, EVAR)
C IOFLAG = IBSET(IOFLAG,20)
IOFLAG = IBSET(IOFLAG,22)
IF (COMM.EQ. 'GEOH' .OR. COMM.EQ. 'GEOMET') THEN
  IOFLAG = IBSET(IOFLAG,11)
  IOFLAG = IBSET(IOFLAG,12)
ELSE IF (COMM.EQ. 'HSTRS') THEN
  IOFLAG = IBSET(IOFLAG,21)
ELSE IF (COMM.EQ. 'ISTRH') THEN
  IOFLAG = IBSET(IOFLAG,1)
ELSE IF (COMM .EQ. 'HSTRH') THEH
  IOFLAG = IBSET(IOFLAG,4)
  IOFLAG = IBSET(IOFLAG,22)
  IOFLAG = IBSET(IOFLAG,24)
ELSE IF (COMM.EQ. 'ISTRH') THEN
  IOFLAG = IBSET(IOFLAG,2)
ELSE IF (COMM.EQ. 'DISP' .OR. COMM.EQ. 'DISPLA') THEN
  IOFLAG = IBSET(IOFLAG,7)
  IOFLAG = IBSET(IOFLAG,26)
ELSE IF (COMM.EQ. 'REACTI') THEN
  IOFLAG = IBSET(IOFLAG,10)
ELSE IF (COMM.EQ. 'EQUILI') THEN
  IOFLAG = IBSET(IOFLAG,27)
  IOFLAG = IBSET(IOFLAG,6)
  IOFLAG = IBSET(IOFLAG,9)
ELSE IF (COMM.EQ. 'COBRE') THEN
  IOFLAG = IBSET(IOFLAG,28)
ELSE IF (COMM.EQ. 'ELEH' .OR. COMM.EQ. 'ELEMEB') THEN
  READ (BUFF, *, EHD=180) ISTART
  IEND = ISTART
  IHTR = 1
  IFG = 1
ELSE IF (COMM.EQ. 'TO') THEN
  READ (BUFF, *, EHD=180) IEND
ELSE IF (COMM.EQ. 'BY') THEN
  READ (BUFF, *, EHD=180) IHTR
ELSE IF (COMM.EQ. 'NODE' .OR. COMM.EQ. 'NODES') THEN
  READ (BUFF, *, EHD=180) ISTART
  IEND = ISTART
  IHTR = 1
  IFG = 2
ELSE IF (COMM.EQ. 'DNE') THEN
  GO TO 130
ELSE IF (COMM.EQ. 'COMRES') THEN
  GO TO 110
ELSE IF (COMM.EQ. 'END') THEN
  RETURN
ELSE
  ICOMM = 1
  GO TO 170
ENDIF
C
IF ($ .NE. 0) THEN
  GO TO 120
ELSE
  GO TO 110
ENDIF
C
130 CONTINUE
IF (IFG .EQ. 1) THEN
  DO KK = ISTART, IEND, IHTR
    IOEL(KK) = 1
  END DO
GO TO 100
ELSE IF (IFG .EQ. 2) THEN
   DO KK = ISTART, IEND, IHTR
      IOBCD(KK) = 1
   END DO
GO TO 100
ENDIF

CONTINUE
ICOMM = 2
CONTINUE
RETURN
CONTINUE
CALL ERRORS (3, 1, 'IOOUT ')
END

C********** EBORAS **********

C SUBROUTINE EBORAS

C PROGRAM:
C EBORAS (EBCDIC or ASCII) determines the integer value for
C the control characters used by the command processor COMPRO.
C This program also determines whether the machine used is an
C ASCII or EBCDIC machine.
C
C =*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*

C IMPLICIT INTEGER (A-2)
C ..SWITCHES: RESUMB=100:10,FORMAT=900:10
C ..SWITCHES:
C****************************************************************

A = ICHAR('a')
AU = ICHAR('A')

---- A = integer equivalent of the lower case A
---- AU = integer equivalent of the upper case A

---- IUPER is a number that must be added to the integer equivalent
of the lower case characters in order to convert to upper case.
For all machines (ASCII or EBCDIC) the increment between the
integer equivalents of the lower case and upper case characters
is fixed, hence IUPER = AU - A

IUPER = AU - A

Z = ICHAR('z')
SPACE = ICHAR(' ')
COMMA = ICHAR(',')
ZERO = ICHAR('0')
HIE = ICHAR('9')
DOT = ICHAR(',')
MINUS = ICHAR('--')
PLUS = ICHAR('+')
EXCL = ICHAR('!')
QUOTE = ICHAR('"')
EQUAL = ICHAR('=')
E = ICHAR('e')
D = ICHAR('d')
K = 0

C
   RETURN
END

C ================================================== COMPRO ==================================================
C
C INCLUDE (PROCESS)
SUBROUTINE COMPRO(NEXT,SVAR)
C
C PROGRAM:
C COMPRO (Command Processor) is used extract the command which
C are recognizable by NLRC3D I/O module from each record (BUFFER)
C of the input file. The command and the parameters associated
C with it are the returned to the calling program.
C
C ON ENTRY:
C BUFFER = character string 80 bytes long which contains one
C record of the input file (passed by COMMON).
C
C ON RETURN:
C COMM = character string 4 bytes long which contains the
C first four characters of the command.
C NEXT = is an integer which tells the calling routine whether
C the input file record BUFFER contains additional
C commands or not. When additional commands are detected
C the calling routine may repeat the call to
C COMPRO with the same BUFFER in order to extract the
C additional commands.
C =0; no additional commands detected
C =1; additional commands detected
C
C ON RETURN:
C COMM = character string 4 bytes long which contains the
C first four characters of the command.
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C...SWITCHES: REHUMB=100:10,FGRHAT=900:10
C...SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ERRORS
C
C******************************************************************************
INTEGER SPACE, COMMA, ZERO, BIBE, DOT, PLUS, EXCL, A, Z, QUOTE, EQUAL
INTEGER N, D
CHARACTER*80 BUFFER, BUFF
CHARACTER*6 COMM
CHARACTER*1 CCOKH
COMMON/COMP1/SPACE, COMMA, ZERO, BIBE, DOT, MINUS, EXCL, A, Z, QUOTE, EQUAL,
IF (BCHAR.EQ.EQUAL .OR. BCHAR.EQ.COMMA) BCHAR = SPACE

IF (K .EQ. 80) THEN
  IF (ICHTR .NE. 1) COMM = 'COMMEB'
  GO TO 110
ELSE IF (BCHAR .EQ. EXCL) THEN
  HEXT = 0
  IF (ICHTR .NE. 1) COMM = 'COMMEB'
  GO TO 110
ELSE IF (BCHAR.EQ.SPACE .AND. IHUM.EQ.1) THEN
  IHUM = 0
  BVAR = BVAR + 1
  K1 = K1 + 1
  BUFF(K1:K1) = '  '
ELSE IF (BCHAR.EQ.SPACE .AND. ICHTR.EQ.1) THEN
  ISPACE = 1
  K1 = XI + 1
  BUFF(K1:K1) = '  '
ELSE IF (BCHAR.EQ.QUOTE .OR. IQUOTE.EQ.1) THEN
  IF (BCHAR.EQ.QUOTE) THEN
    IF (IQUOTE.EQ.1) THEN
      IQUOTE = 0
      HVAR = HVAR + 1
    ELSE IF (IQUOTE.EQ.0) THEN
      IQUOTE = 1
    ENDIF
  ELSE IF (IQUOTE.EQ.0) THEN
    IQUOTE = 1
  ENDIF
ENDIF

C ----- CONVERT LOWER CASE CHARACTERS TO UPPER CASE
C ----- ICOMM IS THE INTEGER EQUIVALENT OF THE UPPER CASE CHARACTER
C
K1 = K1 + 1
IF (BCHAR.GE.A .AND. BCHAR.LE.Z) THEN
  ICOMM = BCHAR + IUPER
ELSE
  ICOMM = BCHAR
ENDIF
CCOMM = CHAR(ICOMM)
BUFF(X1:X1) = CCOMM
ELSE IF (BCHAR.EQ.ZERO .AND. BCHAR.LE.NINE) THEN
  IF (ISPACE.EQ.1) THEN
    ISUM = 1
    K1 = XI + 1
    BUFF(K1:K1) = BUFFER(K:K)
  ELSE
    ICOUNT = ICOUNT + 1
    IF (ICOUNT.LE.6) COMM(ICOUNT:ICOUNT) = BUFFER(K:K)
  ENDIF
ELSE IF (BCHAR.EQ.ZERO .AND. BCHAR.LE.NINE) THEN
  IF (ISPACE.EQ.1) THEN
    ISUM = 1
    K1 = XI + 1
    BUFF(K1:K1) = BUFFER(K:K)
  ELSE
    ICOUNT = ICOUNT + 1
    IF (ICOUNT.LE.6) COMM(ICOUNT:ICOUNT) = BUFFER(K:K)
  ENDIF
ELSE IF (ISPACE.EQ.1 .AND. BCHAR.LE.SPACE) THEN
  IF (ISUM.EQ.1) THEN
    IF (BCHAR.EQ.1 .OR. BCHAR.EQ.D) THEN
      ICOMM = BCHAR + IUPER
    ELSE
      ICOMM = BCHAR
    ENDIF

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
EHDIF
CCOMM = CHAR(ICOMM)
IF (CCOMM.EQ.'E' .OR. CCOMM.EQ.'D') THEN
   K1 = K1 + 1
   BUFF(K1:K1) = CCOMM
ELSE
   CALL ERRORS (1, 1, 'COMPRO')
ENDIF
ELSE
   NEXT = 1
   N = N - 1
   GO TO 110
ENDIF
ELSE IF (NCHAR .EQ. SPACE) THEN
   ICHTR = 1
   ICOUHT = ICOUHT + 1
ENDIF
C ---- CONVERT LOWER CASE CHARACTERS TO UPPER CASE
C ---- ICOMM IS THE INTEGER EQUIVALENT OF THE UPPER CASE CHARACTER
IF (NCHAR.GE.A .AND. NCHAR.LE.Z) THEN
   ICOMM = NCHAR + IUPER
ELSE
   ICOMM = NCHAR
ENDIF
CCOMM = CHAR(ICOMM)
IF (ICOUHT .LE. 6) COMM(ICOUHT:ICOUHT) = CCOMM
ENDIF
GO TO 100
C
110 CONTINUE
IF (HEXT .EQ. 0) K = 0
RETURN
END
C
C ===================================================:==============
C INCLUDE (PROCESS)
SUBROUTINE IHABDY (PROPER, IOUT, TERROR, NTRIM)
C
C PROGRAM:
C IHABDY PROCESSES INPUT ROUTINE FOR nrc3d. THIS PROGRAM READS
C EACH RECORD OF THE INPUT FILES AND EXTRACTS THE COMMAND BY
C CALLING SUBROUTINE COMPR0. THE EXTRACTED COMMANDS ARE THEN
C PROCESSED APPROPRIATELY.
C
C ENTRY:
C IOUT = output device number
C
C ===================================================:==============
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
COMPRO ERRORS

REAL*8 LTHICK
CHARACTER*80 BUFFER,BUFF
CHARACTER*6 COMM
COMM/LAYER/LXLYFD(320,8000)
COMM/INPUT/E/NELEM,NEIG,WHDF,WLINC,MTIT,IFLAG1,IFLAG2,IDIM,
1 EBOD,NCOLG,NFREH
COMM/ISCR11/FRACT(10),WLINC(10),LCODE,INCPR
COMM/ISCR4/IFPOEL(6,6000)
COMM/COMP2/COMM,BUFFER,BUFF
COMM/IALORTH/IFLAG5
DIMENSION ELYMAT(320),PROPER(25,10)
LDEV11 = 11

ALL THE DIFFERENT MATERIALS ARE READ AS A GROUP ONCE TIME
AND MATERIAL NUMBERS ASSIGNED TO THEM SEQUENTIALLY IN
RESPECTIVE OF AN ELEMENT/ELEMENT GROUP REQUIRING ANY PARTICULAR ELEMENT.
MATNUM = 0

FOR EACH ELEMENT OR GROUP OF ELEMENTS....
THE NUMBER OF LAYERS(DEFAULT ONE) IN THE ELEMENT(S) AND
THE FOLLOWING MUST BE SPECIFIED FOR EACH LAYER(NO DEFAULT)

LAYER THICKNESS ONE CONSTANT VALUE OR 4 CORNER VALUES
DISTANCE Z FROM ELEMENT NODE ONE CONSTANT VALUE OR 4 CORNER VALUES
INTEGRATION POINTS IN THREE LOCAL DIRECTIONS
DIRECTION COSINES FOR MATERIAL ORIENTATION(FOR STEEL LAYERS MUST)
MATERIAL PROPERTY NUMBER FOR GIVEN LAYER

100 CONTINUE
LTHICK = 0.
ZI = 0.
LAYNUM = 0
N = 0
ISTART = 0
IEBD = 0
IHTR = 1

DO 110 IK = 1, 320
ELYMAT(IK) = 0.0
110 CONTINUE
IF (0 .NE. 0) GO TO 130

DO 120 CONTINUE
READ (LDEV11, '(*80)', END=140) BUFFER
120 CONTINUE

CALL COMPRO (N, SVAR)

IF (COMM.EQ.'ELEMEB' .OR. COMM.EQ.'ELEM') THEN
  READ (BUFF, *, END=180) ISTART
  IESD = ISTART
  ITR = 1
ELSE IF (COMM.EQ.'TO') THEN

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ELSE IF (COMM .EQ. 'BY') THEN
  IFLAGS = 0
ELSE IF (COMM .EQ. 'SECAHT') THEN
  IFLAGS = 1
ELSE IF (COMM .EQ. 'TAHGEH') THEN
  IFLAG5 = 0
ELSE IF (COMM .EQ. 'MATERI' OR COMM .EQ. 'MAT') THEN
  MATHUM = MATHUM + 1
  IF (MATHUM .GT. 10) CALL ERRORS (3, 1, 'MATHUM')
ELSE IF (COMM .EQ. 'COHC'RE' OR COMM .EQ. 'COHC') THEN
  PROPER(2E, MATHUM) = 1
ELSE IF (COMM .EQ. 'STEEL' OR COMM .EQ. 'REIHFO') THEN
  PROPER(2E, MATHUM) = 0
ELSE IF (COMM .EQ. 'ELASTI' OR COMM .EQ. 'ISOTRO') THEN
  PROPER(2E, MATHUM) = 2
ELSE IF (COMM .EQ. 'E' OR COMM .EQ. 'STIFF') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'E' OR COMM .EQ. 'POISSO') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'ATEMP' OR COMM .EQ. 'THERMA') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'PHO' OR COMM .EQ. 'WEIGHT') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'FC' OR COMM .EQ. 'COMP') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'FT' OR COMM .EQ. 'TEHS') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'EPSC') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'EPSU') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'BSHR' OR COMM .EQ. 'SHEAR') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'GP' OR COMM .EQ. 'ENERGY') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'BUFF' OR COMM .EQ. 'POISSIS') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'BTA') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'FCSTI') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'ATHETA' OR COMM .EQ. 'THRESH') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'TSTIFG') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'CSOFTH') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'YIELD') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'AHARD') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
ELSE IF (COMM .EQ. 'EPSBRK') THEN
  READ (BUFF, *, END=180) PROPER(2E, MATHUM)
C

ELSE IF (COMM.EQ. 'INLAY' OR COMM.EQ. 'ILAYER') THEN
READ (BUFF, *, END=180) ELYMAT(320)
LAYHUM = 0
ELSE IF (COMM.EQ. 'ILMAT') THEN
LAYHUM = LAYHUM + 1
IF (LAYHUM.GT.15) CALL ERRORS (47, 1, 'ELEMENT')
INDEX = 21*(LAYHUM-1) + 1
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'ORIENT') THEN
INDEX = 21*(LAYHUM-1) + 10
READ (BUFF, *, END=180) (ELYMAT(ICOUNT)), ICOUNT = INDEX + 8
ELSE IF (COMM.EQ. 'LTHICK') THEN
INDEX = 21*(LAYHUM-1) + 2
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'TL1') THEN
INDEX = 21*(LAYHUM-1) + 3
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'TL2') THEN
INDEX = 21*(LAYHUM-1) + 4
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'TL3') THEN
INDEX = 21*(LAYHUM-1) + 5
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'Z1') THEN
INDEX = 21*(LAYHUM-1) + 6
READ (BUFF, *, END=180) ZI
ELSE IF (COMM.EQ. 'Z2') THEN
INDEX = 21*(LAYHUM-1) + 7
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'Z3') THEN
INDEX = 21*(LAYHUM-1) + 8
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'Z4') THEN
INDEX = 21*(LAYHUM-1) + 9
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'NIPIT') THEN
INDEX = 21*(LAYHUM-1) + 10
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'NIPETA') THEN
INDEX = 21*(LAYHUM-1) + 19
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'NIPSI') THEN
INDEX = 21*(LAYHUM-1) + 20
READ (BUFF, *, END=180) ELYMAT(INDEX)
ELSE IF (COMM.EQ. 'EHD') THEN
GO TO 190
ELSE IF (COMM .EQ. 'DONE') THEN
   JCOMM = 1
   GO TO 150
ENDIF

C
IF (H .NE. 0) THEN
   GO TO 130
ELSE
   GO TO 120
ENDIF

C
140 CONTINUE
JCOMM = 2

160 CONTINUE
DO 170 K = ISTART, IEND, IHTR

  ELLYFD(320,K) = ELYMAT(320)
  HLAYS = IHT(ELLYMAT(320))
  IF (HLAYS .EQ. 0) THEN
     HLAYS = 1
  ELLYFD(320,K) = 1
  ENDIF
  DO 160 ILY = 1, HLAYS
     ICTH = 21*(ILY-1) + 1
     ELLYFD(ICTH,K) = ELYMAT(ICTH)
     ELLYFD(ICTH+1,K) = ELYMAT(ICTH+1)
     ELLYFD(ICTH+2,K) = ELYMAT(ICTH+2)
     ELLYFD(ICTH+3,K) = ELYMAT(ICTH+3)
     ELLYFD(ICTH+4,K) = ELYMAT(ICTH+4)
     ELLYFD(ICTH+5,K) = ELYMAT(ICTH+5)
     ELLYFD(ICTH+6,K) = ELYMAT(ICTH+6)
     ELLYFD(ICTH+7,K) = ELYMAT(ICTH+7)
     ELLYFD(ICTH+8,K) = ELYMAT(ICTH+8)
     ELLYFD(ICTH+9,K) = ELYMAT(ICTH+9)
     ELLYFD(ICTH+10,K) = ELYMAT(ICTH+10)
     ELLYFD(ICTH+11,K) = ELYMAT(ICTH+11)
     ELLYFD(ICTH+12,K) = ELYMAT(ICTH+12)
     ELLYFD(ICTH+13,K) = ELYMAT(ICTH+13)
     ELLYFD(ICTH+14,K) = ELYMAT(ICTH+14)
     ELLYFD(ICTH+15,K) = ELYMAT(ICTH+15)
     ELLYFD(ICTH+16,K) = ELYMAT(ICTH+16)
     ELLYFD(ICTH+17,K) = ELYMAT(ICTH+17)
     ELLYFD(ICTH+18,K) = ELYMAT(ICTH+18)
     ELLYFD(ICTH+19,K) = ELYMAT(ICTH+19)
     ELLYFD(ICTH+20,K) = ELYMAT(ICTH+20)

160 CONTINUE
170 CONTINUE
IF (JCOMM .EQ. 2) GO TO 180
GO TO 100
180 CONTINUE
CALL ERRORS (3, 1, 'TRAGEDY')
190 CONTINUE
RETURN

C INCLUDE (PROCESS)
C SUBROUTINE IOHIST(IOUT,TERROR)
C
C INCLUDE (PROCESS)
C SUBROUTINE IOHIST(IOUT,TERROR)

C PROGRAM:

C I IODIST PROCESSES INPUT ROUTINE FOR ELRC3D. THIS PROGRAM READS

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
EACH RECORD OF THE INPUT FILE AND EXTRACTS THE COMMAND BY CALLING SUBROUTINE COMPO. THE EXTRACTED COMMANDS ARE THEN PROCESSED APPROPRIATELY.

ENTRY:

IOUT = output device number

IMPLICIT REAL*8 (A-H,O-Z)

SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

CHARACTER*80 BUFFER,BUFF
CHARACTER*6 COMM
COMMON/COMP2/COMM,BUFFER,BUFF
COMMON/PATH,HISTX(10,30),HISTY(10,30),IPATH(IO)

READ ONE LINE OF THE INPUT FILE AND STORE IN BUFFER

LDEV11 = 11
N = 0
ICOUNT = 0
HNUM = 3

IF (0 .GE. 0) GO TO 110
CONTINUE
READ (LDEV11, '(A80)', END=130) BUFFER

EXTRACT THE FIRST SIX CHARACTERS OF EACH COMMAND IN THE BUFFER AND PLACE ALL VARIABLES ASSOCIATED WITH EACH COMMAND IN THE INTERNAL FILE BUFF.

110 CONTINUE
CALL COMPO (N, HVAR)

IF (COMM .EQ. 'POINTS') THEN
HNUM = HNUM + 1
ICOUNT = 0
READ (BUFF, *, END=120) IPATH(HNUM)
ELSE IF (COMM .EQ. 'HAXIS') THEN
ICOUNT = ICOUNT + 1
READ (BUFF, *, END=120) HISTX(HNUM,ICOUNT)
ELSE IF (COMM .EQ. 'HYAXIS') THEN
IF (HISTY(HNUM,ICOUNT) .LT. 0.0) CALL ERRORS (4, 1, 'LOHIST')
READ (BUFF, *, END=120) HISTY(HNUM,ICOUNT)
ELSE IF (COMM .EQ. 'END') THEN
RETURN
ELSE
GO TO 140
ENDIF
IF (N .EQ. 0) THEN
GO TO 100
ELSE
GO TO 110

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ERDFP
120 CONTINUE
CALL ERRORS (4, 1, 'LOCIST')
130 CONTINUE
140 CONTINUE
RETURN
END
C=========================================================================
C INCLUDE (PROCESS)
C SUBROUTINE IOCHST(LAMINA, IOUT, IERROR, ICOMM)
C
C PROGRAM:
C IOCHST PROCESSES INPUT ROUTINE FOR mrc3d. THIS PROGRAM READS
C EACH RECORD OF THE INPUT FILE AND EXTRACTS THE COMMAND BY
C CALLING SUBROUTINE COMPRO. THE EXTRACTED COMMANDS ARE THEN
C PROCESSED APPROPRIATELY.
C
C ENTRY:
C IOUT = output device number
C
C=========================================================================
C IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C COMPRO ERRORS
C
C*****************************************************************************
CHARACTER*80 BUFFER,BUFF
CHARACTER*6 COMM
COMM/COMPRO/COMM,BUFFER,BUFF
DIMENSION LAMINA(6000,2)
LDEVII = 11
C
C...... FOR EACH ELEMENT OR GROUP OF ELEMENTS....
C
C THE NUMBER OF LAYERS (DEFAULT ODE) IN THE ELEMENT(S) AND
C
C
C = 0
ISTART = 0
IEND = 0
ISTR = 1
ICOUNT = 0
C
C 100 CONTINUE
IF ( = .NE. 0) GO TO 120
110 CONTINUE
READ (LDEVII, '(A80)', END=180) BUFFER
C
C ---- EXTRACT THE FIRST SIX CHARACTERS OF EACH COMMAND IN THE BUFFER
C AND PLACE ALL VARIABLES ASSOCIATED WITH EACH COMMAND IN THE
C  INTERNAL FILE BUFF.
C
120 CONTINUE
CALL COMPNG (X, NVAR)
C
IF (COMM .EQ. 'STACK') THEN
  ISTART = 0
  IEND = 0
  ITR = 1
ELSE IF (COMM .EQ. 'ELEM' .OR. COMM .EQ. 'ELEM') THEN
  READ (BUFF, *, END=160) ISTART
  IEND = ISTART
  ITR = 1
ELSE IF (COMM .EQ. 'TO') THEN
  READ (BUFF, *, END=160) IEND
ELSE IF (COMM .EQ. 'BY') THEN
  READ (BUFF, *, END=160) ITR
ELSE IF (COMM .EQ. 'ADD') THEN
  READ (BUFF, *, END=160) IEND
  ICOUNT = ICOUNT + 1
  LAMINA(ICOUNT,1) = ISTART
  LAMINA(ICOUNT,2) = IEND
ELSE IF (COMM .EQ. 'END') THEN
  ICOMM = 0
  GO TO 170
ELSE IF (COMM .EQ. 'STOP') THEN
  ICOMM = 1
  GO TO 130
ELSE IF (COMM .EQ. 'DONE') THEN
  ICOMM = 1
  GO TO 100
ENDIF
C
IF (N .NE. 0) THEN
  GO TO 120
ELSE
  GO TO 110
ENDIF
C
130 CONTINUE
DO 160 K = ISTART, IEND, ITR
  IF (K .EQ. IEND) GO TO 140
  ICOUNT = ICOUNT + 1
  LAMINA(ICOUNT,1) = K
  LAMINA(ICOUNT,2) = K + ITR
140 CONTINUE
150 CONTINUE
  IF (ICOMM .EQ. 2) GO TO 160
  GO TO 100
160 CONTINUE
CALL ERRORS (3, 1, 'IOCHST')
170 CONTINUE
RETURN
180 CONTINUE
ICOMM = 2
END
C******************************************************************************
C INCLUDE(PROCESS)
SUBROUTINE CNPROP(LAMINA, IOUT, IERROR)
IMPLICIT REAL*8 (A-H.O-Z)
C...SWITCHES; RENUMB=100:10, FORMAT=900:10
C...SWITCHES;
C******************************************************************************

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

REAL*4 XYZ, THICK1, THICK2
COMMON/MPCS/CDEFMP(40000), ALAXM(40000), MPCDOF(40000),
$ MPMADR(2,5000), MNPC, MPCTP, MAXMPC
COMMON/LAYERA/ELYPFC(320,6000)
COMMON/SAFEP(60000), IDOF(60000), JDIAG(60000)
COMMON/INPUT2/XYZ(3, 10000)
COMMON/CORES/ICHECK(60000)
COMMON/PROPS/PROPER(25, 16)
DIMENSION LAMINA(6000, 2), THICK1(9), THICK2(9)

ILINK = 0
MPC = 0
MPCTP = 0
DO KM = 1, 6000
   ICHECK(KM) = 0
END DO
DO 220 ICOUNT = 1, 5000
   IF (LAMINA(ICOUNT,1) .NE. 0) THEN
      IELEM1 = LAMINA(ICOUNT,1)
      IELEM2 = LAMINA(ICOUNT,2)
      CALL ELINFO (IELEM1, ITYPE1, NHEL1, IFLAG1, ISTART, LINES)
      CALL ELINTM (IELEM1, IDENT1, INTCOD, NIPXI, NIPETA, NIPS1
      1, MATHUM, THICK1)
      CALL ELINFO (IELEM2, ITYPE2, NHEL2, IFLAG2, ISTART, LINES)
      CALL ELINTM (IELEM2, IDENT2, INTCOD, NIPXI, NIPETA, NIPS1
      1, MATHUM, THICK2)
      IF (ITYPE1.EQ.ITYPE2 .AND. IFLAG1.EQ. IFLAG2) THEN
         DO 210 ND1 = 1, HHEL1
            NODE1 = HOP(ND1, IELEM1)
            TB1 = THICK1(NODE1)
            X1 = XYZ(1, NODE1)
            Y1 = XYZ(2, NODE1)
            Z1 = XYZ(3, NODE1)
            DIST = 1.E30
            DIST1 = 1.E30
            NC = 0
            TH2 = 0.0
         DO 110 ND2 = 1, HHEL2
            NODE2 = HOP(ND2, IELEM2)
            X2 = XYZ(1, NODE2)
            Y2 = XYZ(2, NODE2)
            Z2 = XYZ(3, NODE2)
            DIST = DSQRT((X1-X2)**2+(Y1-Y2)**2+(Z1-Z2)**2)
            IF (DIST.LE. DIST1) THEN
               NC = NODE2
               TH2 = THICK2(NODE2)
               DIST1 = DIST
            ENDIF
         CONTINUE
      ENDIF
   ELSE
      IELEM1 = (MPCTP-1)*6 + 3
      IELEM2 = (MPCTP-1)*6 + 3
      IF (ICHECK(IELEM1).EQ.1 .AND. ICHECK(IELEM2).EQ.1
GO TO 120

ICHECK(IDENT1) = 1
ICHECK(IDENT2) = 1

IDENT1 = (NODER-1)*6 + 3
IDENT2 = (NODER-1)*6 + 3
IF (IDOF(IDENT1).GT.0 .OR. IDOF(IDENT2).GT.0) THEN
  IDOF(IDENT1) = 1
  IDOF(IDENT2) = 1
GO TO 120
ENDIF

ILINK = ILINK + 1
NMC = NMC + 1
MPCD = MPCD + 1
MPCDP(MPCD) = 0
COEPNR(MPCD) = 0.0
MPCDR(1,IMCN) = MPCD
MPCDR(2,IMCN) = 3
MPCD = MPCD + 1
MPCDP(MPCD) = IDENT1
COEPNR(MPCD) = 1.0
MPCD = MPCD + 1
MPCDP(MPCD) = IDENT2
COEPNR(MPCD) = -1.0

GO TO 120

CONTINUE

IDENT1 = (NODER-1)*6 + 1
IDENT2 = (NODER-1)*6 + 5
IDENT3 = (NODER-2-1)*6 + 1
IDENT4 = (NODER-2-1)*6 + 5

IF (ICHECK(IDENT1).EQ.1 .AND. IDCHECK(IDENT2).EQ.1
  .AND. IDCHECK(IDENT3).EQ.1 .AND. IDCHECK(IDENT4)
  .EQ.1) GO TO 160

ICHECK(IDENT1) = 1
ICHECK(IDENT2) = 1
ICHECK(IDENT3) = 1
ICHECK(IDENT4) = 1

IF (IDOF(IDENT1).GT.0 .AND. IDOF(IDENT2).GT.0
  .AND. IDOF(IDENT3).GT.0 .AND. IDOF(IDENT4)
  .GT.0) GO TO 160

IREST = 0
IF (IDOF(IDENT1).GT.0) IREST = 1
IF (IDOF(IDENT2).GT.0) IREST = IREST + 1
IF (IDOF(IDENT3).GT.0) IREST = IREST + 1
IF (IDOF(IDENT4).GT.0) IREST = IREST + 1

ILINK = ILINK + 1
NMC = NMC + 1
MPCD = MPCD + 1
MPCDP(MPCD) = 0
COEPNR(MPCD) = 0.0
MPCDR(1,IMCN) = MPCD
MPCDR(2,IMCN) = 5 - IREST
IF (IDOF(IDENT1).GT.0) GO TO 130
MPCD = MPCD + 1
MPCDP(MPCD) = IDENT1
COEPNR(MPCD) = 1.0
CONTINUE

IF (IDOF(IDENT2).GT.0) GO TO 140
MPCD = MPCD + 1
HPCDOF(HPCPDT) = IDEBT2
COEFHP(HPCPBT) = -1.0
CONTINUE
IF (IDOF(IDEBT2) .GT. 0) GO TO 150
140 HPCPBT = HPCPBT + 1
HPCDOF(HPCPBT) = IDEBT3
COEFHP(HPCPBT) = -1.0*/TH1/2.0
CONTINUE
IF (IDOF(IDEBT3) .GT. 0) GO TO 160
HPCPBT = HPCPBT + 1
HPCDOF(HPCPBT) = IDEBT4
COEFHP(HPCPBT) = -1.0*/TH2/2.0
C
C ICOUNT = ICOUNT + 1
160 CONTINUE
IDEBT1 = (B0DE1-1)*6 + 2
IDEBT2 = (B0DE1-1)*6 + 4
IDEBT3 = (B0DE2-1)*6 + 2
IDEBT4 = (B0DE2-1)*6 + 4
C
IF (ICHECK(IDEBT1).EQ.1 .AND. ICHECK(IDEBT2).EQ.1
1 .AND. ICHECK(IDEBT3).EQ.1 .AND. ICHECK(IDEBT4
2 ).EQ.1) GO TO 200
ICHECK(IDEBT1) = 1
ICHECK(IDEBT2) = 1
ICHECK(IDEBT3) = 1
ICHECK(IDEBT4) = 1
C
IF (IDOF(IDEBT1) .GT. 0 .AND. IDOF(IDEBT2) .GT. 0
1 .AND. IDOF(IDEBT3) .GT. 0 .AND. IDOF(IDEBT4
2 .GT. 0) GO TO 200
C
IREST = 0
IF (IDOF(IDEBT1) .GT. 0) IREST = 1
IF (IDOF(IDEBT2) .GT. 0) IREST = IREST + 1
IF (IDOF(IDEBT3) .GT. 0) IREST = IREST + 1
IF (IDOF(IDEBT4) .GT. 0) IREST = IREST + 1
C
ILINK = ILINK + 1
NMPD = NMPD + 1
MPCPBT = MPCPBT + 1
MPCDOF(MPCPBT) = 0
COEFMP(MPCPBT) = 0.0
MPCADR(1,NMPD) = MPCPBT
MPCADR(2,NMPD) = 5 - IREST
IF (IDOF(IDEBT1) .GT. 0) GO TO 170
MPCPBT = MPCPBT + 1
MPCDOF(MPCPBT) = IDEBT1
COEFMP(MPCPBT) = 1.0
CONTINUE
170 IF (IDOF(IDEBT2) .GT. 0) GO TO 180
MPCPBT = MPCPBT + 1
MPCDOF(MPCPBT) = IDEBT2
COEFMP(MPCPBT) = 1.0*/TH1/2.0
CONTINUE
180 IF (IDOF(IDEBT3) .GT. 0) GO TO 190
MPCPBT = MPCPBT + 1
MPCDOF(MPCPBT) = IDEBT3
COEFMP(MPCPBT) = -1.0
CONTINUE
190 IF (IDOF(IDEBT4) .GT. 0) GO TO 200
MPCPBT = MPCPBT + 1
MPCDOF(MPCPBT) = IDEBT4
COEFHP(MPCPNT) = 1.0*TH2/2.0

C 200 CONTINUE
210 CONTINUE
ENDIF
220 CONTINUE
c CALL WCNS
RETURN
END

C ======================== S U L O A D ========================
C
C INCLUDE (PROCESS)

SUBROUTINE SULOAD(IDIM,N,NHDF,ICOMM,LDEV11,LOUT)
C
C PROGRAM:
C SULOAD READS AND STORES THE SURFACE LOADS

C ON ENTRY:
C IDIM = PHYSICAL DIMENSION OF THE PROBLEM (I.E. 2D OR 3D)
C NHDF = NUMBER OF NODE DEGREES OF FREEDOM
C N = 0; BUFFER CONTAINS NO ADDITIONAL COMMANDS
C 1; BUFFER CONTAINS AT LEAST ONE ADDITIONAL COMMAND
C LDEV11 = UNIT NUMBER FOR THE INPUT FILE
C IOUT = UNIT NUMBER FOR THE OUTPUT FILE

C ON RETURN:
C N = 0; BUFFER CONTAINS NO ADDITIONAL COMMANDS
C 1; BUFFER CONTAINS AT LEAST ONE ADDITIONAL COMMAND
C ICOMM = IDENTIFIES THE ACTION TO BE TAKEN WHEN CONTROL IS
C RETURNED TO THE CALLING ROUTINE .
C =0; READ THE NEXT INPUT LINE
C =1; THE CURRENT COMMAND WAS NOT RESOLVED BY IDDIS.
C cALLING ROUTINE SHOULD PROCESS THIS COMMAND
C =2; END OF FILE IS REACHED, DO NOT TRY TO READ ANY
C MORE LINES.

C================================================================
C
IMPLICIT REAL*8 (A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
CIMPRO ERRORS
C
C*----------------------------------------------------------
C CHARACTER*80 BUFFER,BUFF
C CHARACTER*6 COMM
C COMM/COMP2/COMM,BUFFER,BUFF

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
COMM/STLORD/PRES(6000,9),ISULORD(5000)
DIMENSION P(9)

C
ITIME = 0
ICOMM = 0
ISTART = 0
IEED = 0
IHEX = 0
ICHK = 0
P11 = 0.0
P22 = 0.0
P33 = 0.0
P44 = 0.0
PRESUR = 0.0

C
IF (H .GE. 0) GO TO 110
100 CONTINUE
READ (LDEV11, '(A80)', IEND=190) BUFFER
110 CONTINUE
CALL COMPRO (H, HVAR)
C
IF (COMM.EQ.'ELEM'.OR. COMM.EQ.'ELEMEH') THEN
IF (ITIME .EQ. 1) THEN
ASSIGH 120 TO BEIT
ENDIF
120 CONTINUE
READ (BUFF, *, IEND=200) ISTART
ITIME = 1
IEED = ISTART
IHEX = 1
C
C ---- IDCORD IS THE FLAG FOR THE TRANSATION COORDINATE DEFINITION.
C
C = 0; A LOCAL COORDINATE SYSTEM IS NOT DEFINED
C = 1; A LOCAL COORDINATE SYSTEM IS DEFINED
C
C ---- IBCORD IS THE FLAG FOR THE ROTATIONAL COORDINATE DEFINITION.
C
C = 0; A LOCAL COORDINATE SYSTEM IS NOT DEFINED
C = 1; A LOCAL COORDINATE SYSTEM IS DEFINED
C
DO 130 K1 = 1, 9
P(K1) = 0.
130 CONTINUE
ELSE IF (COMM.EQ.'TO') THEN
READ (BUFF, *, IEND=200) IED
ELSE IF (COMM.EQ.'BY') THEN
READ (BUFF, *, IEND=200) IHEX
ELSE IF (COMM.EQ.'PRESUR') THEN
READ (BUFF, *, IEND=200) PRESUR
ELSE IF (COMM.EQ.'PI') THEN
READ (BUFF, *, IEND=200) P11
ELSE IF (COMM.EQ.'P2') THEN
READ (BUFF, *, IEND=200) P22
ELSE IF (COMM.EQ.'P3') THEN
READ (BUFF, *, IEND=200) P33
ELSE IF (COMM.EQ.'P4') THEN
READ (BUFF, *, IEND=200) P44
ELSE IF (COMM.EQ.'TOP') THEN
ICHX = 1
ELSE IF (COMM.EQ.'BOTTOM') THEN
ICHX = -1
ELSE IF (COMM.EQ.'END') THEN
ICOMM = 0
ASSIGH 140 TO EXIT
GO TO 160

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ELSE IF (COMM .EQ. 'COMMON') THEN
  GO TO 100
ELSE
  ICOMM = 1
  ASSIGN 140 TO NEXT
  GO TO 150
ENDIF

IF (P .NE. 0) THEN
  GO TO 110
ELSE
  GO TO 100
ENDIF

CONTINUE
IF (PRESUR .NE. 0.0) THEN
  DO IM = 1, 9
    P(IM) = PRESUR
  END DO
  PRESUR = 0.0
ENDIF
IF (P11 .NE. 0.0 .OR. P22 .NE. 0.0 .OR. P33 .NE. 0.0 .OR. P44 .NE. 0.0) THEN
  P(1) = P11
  P(2) = P22
  P(3) = P33
  P(4) = P44
  P(5) = (P11 + P22) / 2.0
  P(6) = (P22 + P33) / 2.0
  P(7) = (P33 + P44) / 2.0
  P(8) = (P44 + P11) / 2.0
  P(9) = (P11 + P22 + P33 + P44) / 4.0
  P11 = 0.0
  P22 = 0.0
  P33 = 0.0
  P44 = 0.0
ENDIF
DO 180 K1 = ISTART, IEND, ITRA
  DO 170 IDIR = 1, 9
    ISULOD(K1) = ICHK
    IF (P(IDIR) .NE. 0.) PRES(K1, IDIR) = P(IDIR)
  170 CONTINUE
  180 CONTINUE
ICHECK = 0
GO TO NEXT

CONTINUE
ICOMM = 2
RETURN

CALL ERRORS (3, 1, 'SULOAD')
STOP
END

C
C ============== LOAD ==============
C
C INCLUDE (PROCESS)
SUBROUTINE LOAD (N, IERROR)
C
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
PROGRAM

LOAD ASSEMBLES THE LOAD VECTOR BY CONSIDERING THE
EXTERNALLY APPLIED LOADS AND THE GRAVITY LOADS WHICH ARE
SUPERIMPOSED ON THE STRUCTURE.

ARGUMENT LIST

R(I) = LOAD VECTOR TO BE ASSEMBLED
ERROR = ERROR CODE = 0; NO ERROR < 0; ERROR

COMMON BLOCKS

REFER TO THE COMMON BLOCK DISCIPLINIES.

N(I,J) = SHAPE FUNCTION FOR NODE I AT INTEGR. POINT J
WI = GAUSSIAN WEIGHTING FUNCTIONS
GAUSS = X COORDINATE OF THE GAUSSIAN POINTS IN THE ELEM.
W(TI(I)) = SPECIFIC WEIGHT OF MATERIAL I IN THE I DIR.
W(TY(I)) = SPECIFIC WEIGHT OF MATERIAL I IN THE Y DIR.
W(TZ(I)) = SPECIFIC WEIGHT OF MATERIAL I IN THE Z DIR.
THICK = THICKNESS OF THE ELEMENTS FOR PLANE STR & STU

IMPLICIT REAL*8 (A-E-O-Z)
C SWITCHES: REHUB=100:10,FDRKAT=900:10
C SWITCHES:

SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

C ELISHF ELISTM LYIBFO ISH3DG GAUS
C ISHNL SNORM DINVEC JACE3D GETHEK JACNSL
C ISH22O JAC22D
C

REAL*8 XI,ETA,NSI,SI,LTHICK
REAL*4 SHELLZ,THICK
INTEGER ELUM
COMMON/INPUTI/EXI,EBETA,EBPSI,EBIP,EBCOD
COMMON/INPUT2/EPI(20,6000)
COMMON/SHELDM/VECT(3,9,9)
COMMON/INPUT3/WXT(10),WGT(10),WGTZ(10)
COMMON/INPUT4/RI(60000),RY(60000),RZ(60000)
COMMON/INPUT5/THICK(9),IFLAG
COMMON/INPUT7/CHAR(20000),CHAR1(20000)
COMMON/INPUT8/CHAR1(20000),CHAR(20000),CHAR1(20000)
COMMON/INPUT9/CHAR1(20000),CHAR1(20000),CHAR1(20000)
COMMON/INPUT10/CHAR1(20000),CHAR1(20000),CHAR1(20000)
COMMON/SHELZ/ZS(9),D0S(3,3),HLAYRS,MATRL,LYHUM
COMMON/TRANS/DDE(3,3)
COMMON/SURLOAD/PRES(60000,9),ISURLOD(60000)
DIMENSION R(1),DUMMY(3),XI(4),ETA(4),NSI(4),SI(4),SIZ(4)

C ------- FIND THE CONTRIBUTION OF THE GRAVITY WEIGHTS TO 2D ELEMENTS
DO KL = 1, 60000
    AMX(KL) = 0.0
    AMY(KL) = 0.0
    AMZ(KL) = 0.0
END DO
END 240 ELHUM = 1, HELEN

CALL ELINFO (ELNUM, ITYPE, NHEL, IFLAG, ISTART, LINES)
CALL ELINFO (ELNUM, IDEST, INTCOD, HIPX, HIPETA, HIPSI, MATNUM, ZTRICK)
DO 230 LYHUM = 1, HLAYRS
    THICKL = 0.0
    DETJAC = 0.0
    CALL LYINFO (LYNUM, ZTRICK, ZS, MATAL, DCS, NHEL, ELNUM, HIPX, HIPETA, HIPSI)
    IF (ITYPE .GT. 300) THEN
        IF (ITYPE .GT. 0 .OR. IDEST .GT. 0) THEN
            IF (ITYPE .GT. 0 .OR. IDEST .GT. 0) THEN
                VECT(1,1,K1) = DBLE(SHELLX(KP))
                VECT(2,1,K1) = DBLE(SHELLY(KP))
                VECT(3,1,K1) = DBLE(SHELLZ(KP))
                CALL DIRVEC (VECT(1,K1), VECT(2,K1), VECT(3,K1))
            ELSE
                VECT(1,3,K1) = DBLE(SHELLZ(KP))
                VECT(2,3,K1) = DBLE(SHELLX(KP))
                VECT(3,3,K1) = DBLE(SHELLY(KP))
                CALL DIRVEC (VECT(1,K1), VECT(2,K1), VECT(3,K1))
            END IF
        ELSE IF (ITYPE .EQ. 4) THEN
            ISET = 0
            S1P = S1(ISTOPH)
            CALL GETTHK (ISTOPH, ELNUM, NHEL, HIPX, HIPETA, HIPSI)
        END IF
    END IF
    IF (ICODE .LT. 0) THEN
        CALL SHHORM (ELNUM, K1, HHEL, ITYPE, VECT(1,1,K1), VECT(1,2,K1), VECT(1,3,K1))
    ELSE
        VECT(1,3,K1) = DBLE(SHELLZ(KP))
        VECT(2,3,K1) = DBLE(SHELLX(KP))
        VECT(3,3,K1) = DBLE(SHELLY(KP))
        CALL DIRVEC (VECT(1,K1), VECT(2,K1), VECT(3,K1))
    END IF
    THICKL = 1.0
    DETJAC = 0.0
    CALL ISH3DG (ITYPE, HHEL, TERROR)
    ELSE IF (IFLAG .EQ. 4) THEN
        ISET = 0
        S1P = S1(ISTOPH)
        CALL GETTHK (ISTOPH, ELNUM, NHEL, HIPX, HIPETA, HIPSI)
    END IF
    THICKL = 1.0
    DETJAC = 0.0
    CALL ISHSHL (ITYPE, HHEL, TERROR)
    DO 110 IXI = 1, HHEL
        KP = H0P(K1,ELHUM)
        ICODE = 1ARD(ISPH(KP),4)
        IF (ICODE .GT. 0) THEN
            IF (ICODE .GT. 0) THEN
                CALL SHHORM (ELNUM, K1, HHEL, ITYPE, VECT(1,1,K1), VECT(1,2,K1), VECT(1,3,K1))
            ELSE
                VECT(1,3,K1) = DBLE(SHELLZ(KP))
                VECT(2,3,K1) = DBLE(SHELLX(KP))
                VECT(3,3,K1) = DBLE(SHELLY(KP))
                CALL DIRVEC (VECT(1,K1), VECT(2,K1), VECT(3,K1))
            END IF
        ELSE IF (IFLAG .EQ. 4) THEN
            ISET = 0
            S1P = S1(ISTOPH)
            CALL GETTHK (ISTOPH, ELNUM, NHEL, HIPX, HIPETA, HIPSI)
        END IF
    END IF
    THICKL = 1.0
    DETJAC = 0.0
    CALL ISH3DG (ITYPE, HHEL, TERROR)
    ELSE IF (IFLAG .EQ. 4) THEN
        ISET = 0
        S1P = S1(ISTOPH)
        CALL GETTHK (ISTOPH, ELNUM, NHEL, HIPX, HIPETA, HIPSI)
    END IF
END IF

CALL JACSHL (IHGTGP, ELNUM, HNEL, THICK1, DETJAC, DUMMY, ISET, SIP)

C*VOID: IGNORE RECRDEPS
C*VOID: PREFER VECTOR

DO 120 K1 = 1, HHEL
    M1 = BOP(K1, ELNUM)
    RX(M1) = RX(M1) + H(K1, IHGTGP)*WGT1(M1, IHGTUM)*CST
    RY(M1) = RY(M1) + H(K1, IHGTGP)*WGY1(M1, IHGTUM)*CST
    RZ(M1) = RZ(M1) + H(K1, IHGTGP)*WGTZ1(M1, IHGTUM)*CST
120 CONTINUE

C*----- SURFACE LOADS FOR SHELLS - FOLLOWING PORTIONS NOT TESTED.

MIPSIT = SIPSI
    SIPSI = 1
    CALL ISHSSL (ITYPE, HHEL, IERROR)

DO 200 ISI = 1, MIPSIT
    DO 190 IETA = 1, MIPSIT
    DO 180 IXI = 1, MIPSIT
        IHTGPH = (IXI*IETA)*(ISI-1) + IXI*(IETA-1) + IXI
        ICHK = ISULOD(ELNUM)
        IF (LYHUM.GT.1 .AND. ICHK.EQ.(-1)) GO TO 170
        IF (LYHUM.LT.HLAYRS .AND. ICHK.EQ.1) GO TO 170
        IF (ISI.GT.1 .AND. ICHK.EQ.(-1)) GO TO 170
        IF (IFLAG .EQ. 4) THEN
            ISET = 1
            IF (ICHK .EQ. 0 .AND. ICHK.EQ.0) GO TO 170
            CALL GETTHK (IHGTGP, ELNUM, HNEL, THICK1, RAD)
            CALL JACSHL (IHGTGP, ELNUM, HNEL, THICK1, DETJAC, DUMMY, ISET, SIP)
            DORM = DSQRT(DUMMY(1)**2+DUMMY(2)**2+DUMMY(3)**2)
            DETJAC = 2.0+DETJAC/THICK1
            DUMMY(1) = DUMMY(1)/DORM
            DUMMY(2) = DUMMY(2)/DORM
            DUMMY(3) = DUMMY(3)/DORM
            WT = WGT1(IHGTUM)*WETA(IETA)
        DO 160 K1 = 1, HHEL
            CST1 = DETJAC*WT*FRES(ELNUM, K1)*SIP
            M1 = BOP(K1, ELNUM)
            RX(M1) = RX(M1) + H(K1, IHGTGP)*DUMMY(1)
            RY(M1) = RY(M1) + H(K1, IHGTGP)*DUMMY(2)
            RZ(M1) = RZ(M1) + H(K1, IHGTGP)*DUMMY(3)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ \text{AMX}(H1) = \text{AMX}(M1) + \text{THICK}(K1) \times 0.5 \times (\text{DUMMY}(3) \times \text{VECT}(2,3,K1) - \text{DUMMY}(2) \times \text{VECT}(3,3,K1)) \]

\[ \text{AHY}(H1) = \text{AHY}(K1) + \text{THICK}(K1) \times 0.5 \times (\text{DUMMY}(1) \times \text{VECT}(1,3,K1) - \text{DUMMY}(2) \times \text{VECT}(1,3,K1)) \]

\[ \text{AMZ}(H1) = \text{AMZ}(M1) + \text{THICK}(K1) \times 0.5 \times (\text{DUMMY}(2) \times \text{VECT}(1,3,K1) - \text{DUMMY}(2) \times \text{VECT}(2,3,K1)) \]

Amended code:

```fortran
1 DO 220 IHTGPH = 1, HIP
2 CALL GETTHK(IHTGPH, ELHUM, HHEL, THICK1, RAD)
3 CALL JACB2D(IHTGPH, ELHUM, HHEL, TERROR, DETJAC)
4 CST = DETJAC * THICK1 * W(IHTGPH)
```

```fortran
C*VDIR: IGHORE RECRDEPS
C*VDIR: PREFER VECTOR
C
```

```fortran
DO 210 K1 = 1, HHEL
M1 = EPK(K1, ELHUM)
RX(M1) = RX(M1) + RX(K1, IHTGPH) * WGTX(MATHUH) * CST
RY(M1) = RY(M1) + RY(K1, IHTGPH) * WGTY(MATHUH) * CST
```

```fortran
C --- PLACE RX S AND RY S IN THE RIGHT POSITIONS IN THE
C --- LOAD ARRAY.
C IF (IDIM .EQ. 2) THEN
C*VDIR: PREFER VECTOR
C
```

```fortran
DO 260 K = 1, RHODES
K1 = RHPK(K1, ELHUM)
RI(K1) = RI(K1) + RI(K)
R(K2) = R(K2) + RY(K)
```

```fortran
C ELSE IF (IDIM .EQ. 3) THEN
C*VDIR: PREFER VECTOR
C
```

```fortran
DO 260 K = 1, RHODES
K1 = RHPK(K1, ELHUM)
K2 = K1 + 1
K3 = K2 + 1
K4 = K3 + 1
```
KS = K4 + 1
K6 = K6 + 1
R(K1) = R(K1) + RX(K)
R(K2) = R(K2) + RY(K)
R(K3) = R(K3) + RZ(K)
R(K4) = R(K4) + AMX(K)
R(K5) = R(K5) + AMY(K)
R(K6) = R(K6) + AMZ(K)

260 CONTINUE

ENDIF

C ------ FOR THE NODES WHICH ARE SHARED BY SHELL ELEMENTS AND THEIR
C ROTATIONAL DEGREES OF FREEDOM IS ASSEMBLED IN THE LOCAL
C COORDINATES, TRANSFORM THE LOADS TO THE LOCAL COORDINATES OF
C THE NODE.

C IF (IDIM .EQ. 3) THEN
DO 300 K1 = 1, NNODES
   I = NSDF*(K1-1)
   ICODE = IAND(ISPB(K1),256)
   ISPS = IAND(ISPB(K1),4)
   I = I + IDIM
   IF (ISPS.EQ.0 .AND. ICODE.GT.0) THEN
      DC(1,3) = DBLE(SHELLZ(1,K1))
      DC(2,3) = DBLE(SHELLZ(2,K1))
      DC(3,3) = DBLE(SHELLZ(3,K1))
      CALL DIRVEC (DC(1,1),DC(1,2),DC(1,3))
   ENDIF
C*VDIR: PREFER SCALAR
DO 280 K2 = 1, IDIM
   CST = 0.
   DO 270 K3 = 1, IDIM
      IDIR = I + K3
      CST = CST + R(IDIR)*DC(K3,K2)
   CONTINUE
   DUMMY(K2) = CST
270 CONTINUE
280 CONTINUE
C*VDIR: PREFER SCALAR
DO 290 K2 = 1, IDIM
   IDIR = I + K2
   R(IDIR) = DUMMY(K2)
290 CONTINUE
300 CONTINUE

ENDIF
C RETURN
END

* * * MULTILAYER NON-LINEAR ANALYSIS *
* * *

PROPER(I,K) = 'I'TH PROPERTY OF MATERIAL 'K'.
IBIST= TAPE TO SAVE PREVIOUS HISTORY AT EACH
GAUSS POINT.
ITNSPG= TAPE TO SAVE PREVIOUS HISTORY AT EACH
GAUSS POINT.
HISTO(I,J,K) = ARRAY TO SAVE 'J' NO. OF DATA FOR
GAUSS POINT 'I' OF THE CURRENT ELE-
MENT IN LAYER 'K'.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CRACK(I,J,K) = ARRAY TO SAVE 'J' NO. OF CRACK DATA FOR
GAUSS POINT 'I' OF THE CURRENT ELEMENT IN LAYER 'K'.

TESTF(I,J,K) = ARRAY TO SAVE 'J' NO. OF CRACK DATA FOR
GAUSS POINT 'I' OF THE CURRENT ELEMENT IN LAYER 'K'.

#include(ProcessEvent)

subroutine HLPROM(IOUT,ERROR,IMATE)
imPLICIT REAL*8(A-H,D-Z)

...switches: rhumb=100,10,format=900:10
...switches:

subroutines and functions called from this routine

eliefo elieth lyinfo lshshl gtlxk getshk
dtrans ctsstf

integer elnum
real*8 hulthick
real*4 thick
common/elput9/thick(9),iflag
common/inputs/nodes,elem,ndof,blinc,msit,iflag1,iflag2,idim,$
& node,scolor,bfree
common/elcr1/fact(10),blinc(10),ldcosm,iscptr
common/inputs/elipxi,bipeta,bipsi,iepe,biptcm
common/blocks/histo(25,70,15),crack(25,250,15),ihisty,ihistz,$
& ihist2
common/lprop/prop(25,15)
common/layers/thick(9),zg(9),dcs(3,3),elayers,imat,lynum
common/threal/ai1(5000,9),ai2(5000,9),ai(5000,15,27),$
& af(5000,15,27)
dimension dsteel(6,6),icduht(27,15)
do 100 ihath = 1, hathum

if (prop(25,imat) .eq. 1.0) then

concrete material

ei = prop(1,imat)
fc = prop(6,imat)
epsc = prop(7,imat)
ec = fc/epsc
a = ei/ec
if (a .lt. 4./3.) then
write (lout, 900) imate, a, 'elprop conc. error. 1 '
stop
endif

check for the limits of the post-crushing param. 'd'.

d = prop(9,imat)
if (a .le. 2.) then

1 = (1.-e*a)**2.
c2 = 1. + a*(a-2.)
if (.not.(d.le.c2 .and. d.ge.c1)) then
write (lout, 910) imate, a, d,
1 'NLPROP CONC. ERROR 2'
STOP
ENDIF
ENDIF
IF (.GT. 2.) THEN
    IF (.NOT.(D.GE.0. .AND. D.LE.1.)) THEN
        WRITE (ISOUT, 910) IMATE, A, D,
1 'NLPROP CONC. ERROR 3'
STOP
ENDIF
ENDIF

C 100 CONTINUE
C C
C GENERATE ELEMENTS OF THE INITIAL CONSTITUTIVE MATRIX
C D(1,1),D(1,2),D(2,2),D(2,3),D(3,3),D(4,4),D(5,5),D(6,6).
C THE DIRECTION COSINES W.R.T GLOBAL XYZ, WHICH
C INITIALLY IS PUT TO ZERO FOR CONCRETE LAYERS AND IS ALWAYS
C EQUAL TO THE ORIENTATION OF THE STEEL REBARS. THIS IS DONE
C FOR EACH GAUSS POINT OF EACH ELEMENT AND THE RESULTING DATA
C FOR EACH BLOCK OF ELEMENTS IS SAVED IN ARRAY 'HISTO'.
C
C IHISTY = 50
C IHISTL = 60
C IHIST2 = 61
C ITNSPG = 80
C ITNSG1 = 70
C ITNSG2 = 71
C
C REWIND IHISTY
C REWIND IHISTL
C REWIND IHIST2
C REWIND ITNSPG
C REWIND ITNSG1
C REWIND ITNSG2
C DO 240 ELNUM = 1, NELEM
C C
C INITIALIZE 'HISTO' & 'CRACK' FOR EACH BLOCK
C OF ELEMENTS.
C
C CALL ELIHFO (ELNUM, ITYPE, BREL, IFIL, MHIT, MITE)
C CALL ELIHFM (ELNUM, ITYPE, BREL, IFLAG, IMAT, IBPSL, IBPS,
C MATNUM, THICK)
C 1 DO 150 ILYNUM = 1, 15
C 1 DO 140 IDRAWS = 1, 27
C 1 100 ICOUNT(IGAUS,ILYNUM) = 0
C 1 DO 110 IDATA = 1, 70
C 1 HISTO(IGAUS,IDATA,ILYNUM) = 0.0
C 110 CONTINUE
C 1 DO 120 IDATA = 1, 80
C 1 CRACK(IGAUS,IDATA,ILYNUM) = 0.0
C 120 CONTINUE
C 1 DO 130 IDATA = 1, 80
C 1 TESTF(IGAUS,IDATA,ILYNUM) = 0.0
C 130 CONTINUE
C 140 CONTINUE
C 150 CONTINUE
C C
C DO 230 J = 1, NLYRS
C
CALL LYIBFO (J, LTHICK, 2S, MATRL, DC5, BBEL, ELSUM, HIPXI, HIPETA, HIPXI)

NIP = HIPXI*HIPETA*HIPXI
IF (IFLAG .EQ. 4) THEN
  IF (MLAYRS .GT. 1) THEN
    DO KS = 1, 9
      AK1(ELHUM, KS) = 5.0/6.0
      AK2(ELHUM, KS) = 5.0/6.0
    END DO
    CALL ISHSHL (ITYPE, BBEL, TERROR)
    DO KS = 1, BIP
      THICKL = 0.0
      ZSI = 0.0
      THICKE = 0.0
      CALL GTLTK (KS, ELHUM, BBEL, THICKL, ZSI, LTHICK, ZS)
      CALL GETTHK (KS, ELBUM, BBEL, THICKE, RAD)
      AKOHST = (3./2.)*(1.0-(4.0*ZSI*ZSI)/(THICKE*THICKE))
      AF1(ELBUM, J, KS) = AKOHST
      AF2(ELBUM, J, KS) = AKOHST
    END DO
  ELSE
    DO KS = 1, 9
      AK1(ELHUM, KS) = 5.0/6.0
      AK2(ELHUM, KS) = 5.0/6.0
      AF1(ELBUM, 1, KS) = E./6.
      AF2(ELBUM, 1, KS) = E./6.
    END DO
  END IF
END IF

E = PROPER(1, MATRL)
NU = PROPER(2, MATRL)
DO 220 L = 1, BIP

******** STEEL LAYER ********

IF (PROPER(25, MATRL) .EQ. 0.0) THEN

  -------------- STEEL LAYER --------------
  --------------

SAVE D(1,1) AND THEATA=ORIENTATION OF REBARS(RADIANS)

HISTO(L,1,J) = E
HISTO(L,14,J) = E
HISTO(L,13,J) = PROPER(5, MATRL)

CALCULATE AND SAVE THE YIELD STRAIGHT.

EPSYLD = HISTO(L,13,J)/HISTO(L,14,J)
HISTO(L,12,J) = EPSYLD

GENERATE THE CONSTITUTIVE MATRIX FOR THE STEEL
IN THE MATERIAL AXES (DIREC. OF REBARS).
DO 200 IR = 1, 6
   DO 190 IC = 1, 6
      DSTEEL(IR,IC) = 0.0
   CONTINUE
200 CONTINUE
DSTEEL(1,1) = E

ROTATE 'DSTEEL' TO GLOBAL AXES.

CALL DTRANS (DSTEEL, DCS, IDUT)

... STORE STEEL FIBRE DIRECTION

HIST0(L,17,J) = DCS(1,1)
HIST0(L,18,J) = DCS(1,2)
HIST0(L,19,J) = DCS(1,3)

SAVE THE ROTATED CONSTITUTIVE MATRIX IN ARRAY 'CRACK'

CRACK(L,2,J) = DSTEEL(1,1)
CRACK(L,3,J) = DSTEEL(1,2)
CRACK(L,4,J) = DSTEEL(1,3)
CRACK(L,5,J) = DSTEEL(1,4)
CRACK(L,6,J) = DSTEEL(1,5)
CRACK(L,7,J) = DSTEEL(1,6)
CRACK(L,8,J) = DSTEEL(2,1)
CRACK(L,9,J) = DSTEEL(2,2)
CRACK(L,10,J) = DSTEEL(2,3)
CRACK(L,11,J) = DSTEEL(2,4)
CRACK(L,12,J) = DSTEEL(2,5)
CRACK(L,13,J) = DSTEEL(2,6)
CRACK(L,14,J) = DSTEEL(3,1)
CRACK(L,15,J) = DSTEEL(3,2)
CRACK(L,16,J) = DSTEEL(3,3)
CRACK(L,17,J) = DSTEEL(3,4)
CRACK(L,18,J) = DSTEEL(3,5)
CRACK(L,19,J) = DSTEEL(3,6)
CRACK(L,20,J) = DSTEEL(4,1)
CRACK(L,21,J) = DSTEEL(4,2)
CRACK(L,22,J) = DSTEEL(4,3)
CRACK(L,23,J) = DSTEEL(4,4)
CRACK(L,24,J) = DSTEEL(4,5)
CRACK(L,25,J) = DSTEEL(4,6)
CRACK(L,26,J) = DSTEEL(5,1)
CRACK(L,27,J) = DSTEEL(5,2)
CRACK(L,28,J) = DSTEEL(5,3)
CRACK(L,29,J) = DSTEEL(5,4)
CRACK(L,30,J) = DSTEEL(5,5)
CRACK(L,31,J) = DSTEEL(5,6)
CRACK(L,32,J) = DSTEEL(6,1)
CRACK(L,33,J) = DSTEEL(6,2)
CRACK(L,34,J) = DSTEEL(6,3)
CRACK(L,35,J) = DSTEEL(6,4)
CRACK(L,36,J) = DSTEEL(6,5)
CRACK(L,37,J) = DSTEEL(6,6)
HIST0(L,15,J) = 1.0
HIST0(L,16,J) = 1.0
GO TO 210

ENDIF

-----------------------
CONCRETE LAYERS

CONST = E/((1.0-2.0*NU)*(1.0+NU))

CALCULATE D11, D12, D22, D33, D44, D66

HISTO(L,1,J) = CONST*(1.0-NU)
HISTO(L,2,J) = CONST*NU
HISTO(L,3,J) = CONST*(1.0-NU)
HISTO(L,4,J) = CONST*(1.0-NU)
HISTO(L,5,J) = CONST*NU
HISTO(L,6,J) = CONST*NU
HISTO(L,7,J) = E/(2.0*(1.0+NU))
HISTO(L,8,J) = E/(2.0*(1.0+NU))
HISTO(L,9,J) = E/(2.0*(1.0+NU))

HISTO(L,10-18,J) = DIR. COS. FROM GLOBAL XYZ

HISTO(L,10,J) = 0.0
HISTO(L,11,J) = 0.0
HISTO(L,12,J) = 0.0
HISTO(L,13,J) = 0.0
HISTO(L,14,J) = 0.0
HISTO(L,15,J) = 0.0
HISTO(L,16,J) = 0.0
HISTO(L,17,J) = 0.0
HISTO(L,18,J) = 0.0
HISTO(L,19,J) = E
HISTO(L,20,J) = NU

THE COMPRESSION SOFTENING MODEL NEED TO STORE AND UPDATE

THE COMP. STRENGTH FOR EACH ELEMENT, CONCRETE LAYER AT EACH
SAMPLING POINT APPROPRIATELY. SO STORE THIS PARAMETER CORRECTLY.

HISTO(L,58,J) = PROPER(S,MATRL)

INITIALLY THE CONSTITUTIVE MATRIX IN THE GLOBAL
AXES IS THE SAME AS ONE IN THE MATERIAL AXES.

D11, D12, D13, D14, D15, D16

CRACK(L,2,J) = HISTO(L,1,J)
CRACK(L,3,J) = HISTO(L,2,J)
CRACK(L,4,J) = HISTO(L,3,J)
CRACK(L,5,J) = 0.0
CRACK(L,6,J) = 0.0
CRACK(L,7,J) = 0.0

D21, D22, D23, D24, D25, D26

CRACK(L,8,J) = HISTO(L,2,J)
CRACK(L,9,J) = HISTO(L,3,J)
CRACK(L,10,J) = HISTO(L,6,J)
CRACK(L,11,J) = 0.0
CRACK(L,12,J) = 0.0
CRACK(L,13,J) = 0.0

D31, D32, D33, D34, D35, D36

CRACK(L,14,J) = HISTO(L,6,J)
CRACK(L,15,J) = HISTO(L,5,J)
CRACK(L,16,J) = HISTO(L,4,J)
CRACK(L,17,J) = 0.0
CRACK(L,18,J) = 0.0
CRACK(L,19,J) = 0.0

CRACK(L,20,J) = 0.0
CRACK(L,21,J) = 0.0
CRACK(L,22,J) = 0.0
CRACK(L,23,J) = HISTO(L,7,J)
CRACK(L,24,J) = 0.0
CRACK(L,25,J) = 0.0

CRACK(L,26,J) = 0.0
CRACK(L,27,J) = 0.0
CRACK(L,28,J) = 0.0
CRACK(L,29,J) = 0.0
CRACK(L,30,J) = HISTO(L,8,J)
CRACK(L,31,J) = 0.0

CRACK(L,32,J) = 0.0
CRACK(L,33,J) = 0.0
CRACK(L,34,J) = 0.0
CRACK(L,35,J) = 0.0
CRACK(L,36,J) = 0.0
CRACK(L,37,J) = HISTO(L,9,J)
IF (PROPER(28,MATRL) .EQ. 2 ) GO TO 210

CRACK(L,1,J) = 0.0
IF (BLAYRS .GT. 1) CALL CTHSTF (ELNUM, L, J, IOUT,
                                BLAYRS, ICOUNT)

HISTO(L,29,J) = 1. .... STRESS POINT IS PRE-FAILURE

= 2. .... ; ; FAILURE STATE
= 3. .... ; ; POST-FAILURE
=4. .... CRUSHED POINT (STIFFNESS=0.)

INITIALLY ALL POINTS ARE IN THE PRE-FAILURE STATE.

HISTO(L,29,J) = 1.0

HISTO(L,30,J) = 1. .... LOADING OF THE STRESS POINT

= 2. .... UNLOADING ; ; ;
C = 3. .... RELOADING ; ; ;

C INITIALLY ALL POINTS ARE LOADING (STRAINS INCREASING)

C SETUP THE TENSION-STIFFENING LAYER ISF0 IS CONCRETE LAYERS NEXT TO

C STEEL

HISTO(L,30,J) = 1.0

210 CONTINUE
220 CONTINUE

WRITE (ITHIST) ELHUM, J, HIP, ((CRACK(L,K,J), K = 1, 280), L = 1, HIP)
WRITE (ITHSPG) ELHUM, J, HIP, MATHL, ((HISTF(L,K,J), K = 1,
1 ,80), L = 1, HIP), ((HISTO(L,K,J), K = 1, 70), L = 1,
2 HIP), (PROPER(KM,MATRL), KM = 1, 26)
WRITE (ITHST) ELHUM, J, HIP, ((CRACK(L,K,J), K = 1, 280), L = 1, HIP)
WRITE (ITHSTG) ELHUM, J, HIP, MATHL, ((HISTF(L,K,J), K = 1,
1 ,80), L = 1, HIP), ((HISTO(L,K,J), K = 1, 70), L = 1,
2 HIP), (PROPER(KM,MATRL), KM = 1, 26)

230 CONTINUE
240 CONTINUE

RETURN

900 FORMAT (///, 'ERROR', /, 'PARAMETER A FOR ',
1 'CONCRETE MATRL ',12,IX,' IS LESS THAN 4./3.',/, 2
1 IX,'A=',DIO.3)
910 FORMAT (///, 'ERROR', /, 'FOR CONCRETE MATRL: ',
1 'STOPPING PARAMETER,D=',D12.B,IX,' WHICH LIES OUTSIDE ITS ALLOWABLE RANGE')

END

C======================================================================
C INCLUDE(PROCESS)
C
SUBROUTINE CHKRC (INCREM, HIT, TERROR, lOUT, ICHECK)
IMPLICIT REAL*8(A-H,D-Z)
C. . .SWITCHES: RELURB=100: 10,FORMAT=900:10
C...SWITCHES: 
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
REAL*4 THICK, SHELL
REAL*8 LTHICK, W, BZI, C, K, SI
INTEGER ELHUM, ELHUMB
CHARACTER*105,DUMMY
COMMON/INPUT8/ISPBH(10000)
COMMON/SETR/OHELLS(3,10000)
COMMON/SDIDIA/VECT(3,3,9)
COMMON/TRANS/V1(3),V2(3),V3(3)
COMMON/INPUTS/BOIDES,HELEM,BNDL,HIFLC,HEMT,IFLAG, IFLAG2, IDIM,
$ BIRDGE,SCOLOR,BFRES
COMMON/ISIZE/PLACT(10),HILRCH(10),LDCOHT, XCPRTR
COMMON/ISHAP/H(20,27),BZI(20,27),BETA(20,27),HSI(20,27), SI(27)
COMMON/IINPUT/THECK(9),IFLAG
COMMON/IINPUT/H/HEPL,HEPSI,HIPHI,HIP,SITCOD
COMMON/LAYERS/LTHECK(9),ZS(9),DCS(3,3),ELAYRS,MATHL,LYHUM
COMMON/LPROP/PROPER(28,16)
COMMON/ABC/HISTF(27,80,16),ITHSPG,ITHSTG,ITHSTG2
COMMON/BLOCKS/HISTO(27,70,16),CRACK(27,280,16),ITHST,ITHST1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
$ .IHIST2
COMMDS/TSHEAR/AKI(5000,9),AK2(5000,9),AF1(5000,15,27)
$ .IHIST1/STRESS(6),STRAINS(6),DUMMY
COMMDS/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEVS,LDESV,LDEVS
COMMDS/INPUTS/BDP(20,5000)
COMMDS/TSHEAR/E1(5000,9),F1(5000,9)
COMMDS/MATERIAL(5E6,6)
DIMENSION TEMPI(9),TEMP2(9),TEMPS(9),TEMP4(9),AR1(9),AR2(9)
DIMENSION TEMP1B(9),TEMP2B(9),TEMPSB(9),TEMP4B(9),AR1B(9),AR2B(9)
DIMENSION TEMPS(18,27),TEMP6(9),TEMP7(18,27),TEMP8(9),TEMPS(9)
DIMENSION TEMPSB(9),TEMPSB(9),TEMPSB(9),TEMPSB(9)
DIMENSION TEMPS(9),TEMPS(9),TEMPS(9)

C
C SUBROUTINE CONTROLS THE CONVERGENCE PROCEDURE
C AND CALLING OF APPROPRIATE SUBROUTINES.
C
C FOR EACH ELEMENT, LAYER AND INTEGRATION POINT:
C
C CONVER = 1. ........ BOTH 'E' AND 'MU' ARE WITHIN
C THE TOLERANCE, I.E., CONVERGENCE IN CURRENT LAYER
C
C CONVER =999. ........ EITHER 'E' OR 'MU' FOR AT LEAST
C ONE GAUSS POINT VIOLATES THE ALLOW. TOLER
C
C 'HSTIFF' DETERMINES IF STIFFNESS MATRIX FOR AT LEAST
C ONE LAYER OF ELEMENT 'I' IS TO BE UPDATED.
C
C HSTIFF = 0 .... NO STIFFNESS UPDATE NECESSARY.
C
C HSTIFF = 1 .... STIFFNESS UPDATE NECESSARY.
C
C=====================================================================
C
ICHECK = 0
REWIND IHIST2
REWIND IHIST1
REWIND IHIST2
REWIND ITNSPG
REWIND ITNSG1
REWIND ITNSG2
DO 210 ELNUM = 1, HELEM
C
CALL ELISFO (ELEM, ITYPE, MBE, IFLAG, INSTAT, LINES)
CALL ELISTM (ELEM, IDEST, ITCODE, HIPXI, HIP ETA, HIPSI,
1 MATNUM, THICK)
C
CONVER = 1.0
HSTIFF = 0
C
DO IC = 1, 9
AR1(IC) = 0.0
AR2(IC) = 0.0
TEMP1(IC) = 0.0
TEMP2(IC) = 0.0
TEMPS(IC) = 0.0
TEMP4(IC) = 0.0
TEMP6(IC) = 0.0
TEMP8(IC) = 0.0
TEMPSA(IC) = 0.0
TEMPSB(IC) = 0.0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
&TEHPC(IC) = 0.0
&TEHPD(IC) = 0.0
&ARI1B(IC) = 0.0
&ARI2B(IC) = 0.0
&TEMPIB(IC) = 0.0
&TEHP2BCIC) = 0.0
&TEHPBBdC) = 0.0
&TEHP4B(IC) = 0.0
&TEHP6B(IC) = 0.0
&TEHP7B(IC) = 0.0
&TEHP8B(IC) = 0.0
&TEHP9B(IC) = 0.0
&TEHP10B(IC) = 0.0
&TEHP11B(IC) = 0.0
&TEHP12B(IC) = 0.0
&TEHPC(IC) = 0.0
&TEHPD(IC) = 0.0
END DO
DO LYBUM = 1, 15
  DO IC = 1, 27
    TEMP6(LYBUM,IC) = 0.0
    TEMP7(LYBUM,IC) = 0.0
  END DO
END DO
C
DO 170 LYBUM = 1, ELYUMS
  CALL LYIBFO (LYBUM, LTHICK, ZS, MATRL, DCS, BBEL, ELBUM,
                HIPXI, HIPETA, HIPSI)
  C
  C RECOVER ARRAY 'HISTO', 'CRACK', 'TEBSTF' FOR THE CURRENT ELEMENT.
  C THESE ARE REQUIRED FOR THE UPDATING PROCEDURE IN THE CONVERGENCE
  C CRITERION EMPLOYED. STORES ALL THE MATERIAL INFO ABOUT POINT.
  C
  HIP = HIPXI*HIPETA*HIPSI
  C
  IF (ELEUM .NE. ELYUM .OR. LYBUMB .NE. LYBUM .OR. HGAUS .NE. HIP
     ) THEN
    WRITE (*, *) 'ERROR READING IHISTY IN CHKRC'
    WRITE (*, *) 'ELBUM', ELBUM, 'ELBUMB', ELBUMB, 'LYBUM', LYBUM
    STOP
  ENDIF
  C
  C IF (ITBSPG) ELEUM, LYMUM, HGAUS, ((TEBSTF(L,K,LYBUM),K
     = 1,80), L = 1, HGAUS)
  1
  2
  3
  C
  IF (ELEUM .NE. ELYUM .OR. LYBUMB .NE. LYBUM .OR. HGAUS .NE. HIP
     ) THEN
    WRITE (*, *) 'ERROR READING ITBSPG IN CHKRC'
    WRITE (*, *) 'ELBUM', ELBUMB, 'ELSUM', ELSUMB, 'ELSUM', ELSUMB, 'LYBUM'
    STOP
  ENDIF
  C
  IF (ITYPE .LE. 0) GO TO 200

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IF (ITYPE .EQ. 0 .OR. IDENT .EQ. 0) THEN
  IF (ITYPE .GT. 300) THEN
    IF (IFLAG .EQ. 0) THEN
      HCB = 3*HHHEL
      CALL ISH3DC (ITYPE, HHHEL, TERROR)
    ELSE IF (IFLAG .EQ. 4) THEN
      HCB = 3*HHHEL
      CALL ISH3DC (ITYPE, HHHEL, TERROR)
    ENDIF
  ELSE IF (ITYPE .GT. 200) THEN
    HCB = 2*HHHEL
    CALL ISH2DG (ITYPE, HHHEL, TERROR)
  IF (IFLAG .EQ. 3) THEN
    ELSE
  ENDIF
ENDIF

C ITYPE1 = ITYPE
C IDENT1 = IDENT
C****

IF (IFLAG .EQ. 4) THEN
  DO 130 K1 = 1, HHHEL
    KP = NNP(K1,ELNUM)
    ICODE = IAND(ISPB(KP),4)
    IF (ICODE .GT. 0) THEN
      CALL SHHORM (ELHUM, K1, HHHEL, ITYPE, VECT(1,1,K1),VECT(1,2,K1),VECT(1,3,K1))
    ELSE
      VECT(1,3,K1) = DBLE(SHELLZ(1,KP))
      VECT(2,3,K1) = DBLE(SHELLZ(2,KP))
      VECT(3,3,K1) = DBLE(SHELLZ(3,KP))
      CALL DIRVEC (VECT(1,1,K1),VECT(1,2,K1),VECT(1,3,K1))
    ENDIF
  130 CONTINUE
ENDIF

C DO 160 INTGPH = 1, HEPH
THICKE = 0.0
IF (IFLAG .EQ. 4) THEN
  CALL GETTHK (INTGPH, ELNUM, HBEL, THICKE, RAD)
C
C ---- VECTOR V3 RETURNED BY JACSHL IS NORMAL TO MID SURFACE OF THE
C SHELL AT INTEGRATION POINTS
C
ISET = 0
SIP = SZ(INTGPH)
CALL JACSHL (INTGPH, ELNUM, HBEL, THICKE, DETJAC, V3, ISET, SIP)

C ---- THE COORDINATES VECTORS V1 AND V2 ARE EVALUATED BY DIRVEC
C WHICH IS PART OF THE ELEMENT LIBRARY MODULE
C
CALL DIRVEC (V1, V2, V3)
C****
.... CHECK THE STEEL LAYER

IF (PROPER(25, MATRL) .EQ. 0.0) THEN

CHECK THE CONVERGENCE OF THE MATERIAL LAW AT EACH GAUSS POINT OF THE STEEL LAYERS.

TOLER = PROPER(22, MATRL)
IF (TOLER .EQ. 0.0) TOLER = 3.0
CALL CHKSTL (PROPER(1, MATRL), TOLER, NSTIFF, CONVER,
ELNUM, INTOPS, ITYPE, IOUT)
ELSE IF (PROPER(25, MATRL) .EQ. 1.0) THEN

CHECK THE CONVERGENCE OF THE MATERIAL LAW AT EACH GAUSS POINT OF THE CONCRETE LAYERS.

TOLER = PROPER(22, MATRL)
IF (TOLER .EQ. 0.0) TOLER = 3.0
CALL CHKCOH (TOLER, ELNUM, IOUT, PROPER(1, MATRL),
CONVER, NSTIFF, NHEL, INCREM, ITYPE, INTOPS,
ERROR, IFLAG)
ELSE IF (PROPER(25, MATRL) .EQ. 2.0) THEN
CALL J0GET (LDEV1, 96, '(A96)', 5)
ENDIF

... COMPUTE THE TRANSVERSE STIFFNESS CORRECTION FACTOR FOR A SHELL ELEMENT ACCOUNTING FOR ISOTROPIC MATERIAL LAYERS AND CRACKING.

IF (IFLAG .EQ. 4) THEN
IF (LAYERS .GT. 1) THEN

1

THICKL = 0.0
ZSI = 0.0
ACCONS = 0.0
Z1 = 0.0
Z2 = 0.0
CALL GTLTK (INTOPS, ELNUM, NHEL, THICKL, ZSI, 

LTHICK, Z5)
1

IF (NIPS1 .EQ. 1) THEN
ACONS = THICKL/2.0
ELSE IF (NIPS1 .EQ. 2) THEN
IC = INTOPS/(NIPSI*NIPSI)
IP = INTOPS - IC*NIPSI*NIPSI
IF (IC.EQ.0.0.OR.IP.EQ.0.0.AND.IC.EQ.1) THEN

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ACOHST = THICKL*0.67735/2.0
ELSE IF (IC.EQ.1 .OR. IC.EQ.2 .AND. IP.EQ.
0) THEN
ACOHST = THICKL*0.42265/2.0
ENDIF
ELSE IF (HIPSI .EQ. 3) THEN
IC = IHTGPH/(HIPXI*HIPETA)
IP = IHTGPH - IC*HIPXI*HIPETA
IF((IC.EQ.0 .OR. IC.EQ.1 .AND. IP.EQ.0) .OR. IC.EQ.2 .AND. IP.EQ.1) THEN
ACOHST = THICKL*0.7745986/2.0
ELSE IF (IC.EQ.1 .OR. IP.EQ.0 .AND. IC.EQ.2) THEN
ACOHST = THICKL/2.0
ELSE IF (IC.EQ.2 .OR. IP.EQ.0 .AND. IC.EQ.3) THEN
ACOHST = THICKL*0.2254034/2.0
ENDIF
ENDIF

C
DO IRS = 1, 6
DO IPS = 1, 6
DEP(IRS,IPS) = 0.0
END DO
END DO

C
ICODE = 0
IPG = 1
CALL DMATCS (ELHUM, ITYPE, IHTGPH, IFLAG, IOUT
,ICODE, IPG)

C
DEP(6,6) = DEP(6,6)/(AF2(ELHUM,LYHUM,ISTOPH)*
S./6.)
DEP(6,6) = DEP(6,6)/(AF1(ELHUM,LYHUM,ISTOPH)*
S./6.)

C
Z1 = (-E1(ELHUM,IHTGPH)) + ZSI + THICKE/2.0
Z2 = (-F1(ELHUM,IHTGPH)) + ZSI + THICKE/2.0

C
C TRANSVERSE COMPONENTS IN THE X DIRECTION
C
C COMPUTE THE A1 TERM
C
AR1(IHDEX) = DEP(1,1)*(1/3.)*(Z1**3-TEMPA(
INDEX)**3) + AR1(IHDEX) + ARIB(IHDEX)
ARIB(IHDEX) = DEP(1,1)*(1/3.)*((Z1+ACOHST)**3-
Z1**3)

C
C COMPUTE THE G TERM, GBAR TERM AND ACCUMULATE
C
TEMPS(IHDEX) = TEMPC(IHDEX)*(Z1-TEHPA(IHDEX))
2 /THICKE - DEP(1,1)*((Z1**3-TEMPA(INDEX)**3)
2 /6.-Z1-TEMPA(INDEX))
THICKE + TEMPSB(INDEX) + TEMPS(INDEX)

C
COMPUTE THE EG TERM

C IF (PROPER(2B,MATRL) .LE. 0.0) THEN
    TEMP1(Index) = TEMP1(Index) + DEP(6,6)*(Z1 - TEMP1(Index))
    TEMP1B(Index) = DEP(6,6)*ACOHST
    ZZ1 = Z1 + TEMP1(Index)
    ZZ2 = Z1**2 - TEMP1(Index)**2
    ZZ3 = Z1**3 - TEMP1(Index)**3
    ZZ4 = Z1**4 - TEMP1(Index)**4
    ZZ5 = Z1**5 - TEMP1(Index)**5
    T1 = TEMPC(Index)+TEMPC(Index)>XX1
    T2 = (TEMP1(Index)**2+DEP(1,1)*XX1)/4.0
    T3 = (IX6+DEP(1,1))/2.0
    T4 = (TEMP1(Index)**2+II3+DEP(1,1))/6.0
    T5 = TEMPC(Index)+TEMPC(Index)**2+XX1
    T6 = TEMPC(Index)+XX3/3.0
    TEMP3(Index) = TEMP3(Index) + (T1+DEP(1,1)
        +(T6-T6+T2+T3-T4))/DEP(6,6) + TEMP3B(1)
    INDEX
    X1B = ACONST
    X2B = (Z1+ACOHST)**2 - Z1**2
    X3B = (Z1+ACOHST)**3 - Z1**3
    X4B = (Z1+ACOHST)**4 - Z1**4
    X5B = (Z1+ACOHST)**5 - Z1**5
    T1B = TEMPS(LYHUM,ISTGPH)*TEMPS(LYHUM,
        IHTGPH) + XX1B
    T2B = (Z1**4+DEP(1,1)*XX1B)/4.0
    T3B = (IX6+DEP(1,1))/2.0
    T4B = (Z1**2+XX3B+DEP(1,1))/6.0
    T5B = TEMPS(LYHUM,ISTGPH)*Z1**2+XX1B
    T6B = TEMPS(LYHUM,ISTGPH)+XX3B/3.0
    TEMP3B(Index) = (T1B+DEP(1,1)*(T5B-T6B+T2B
        +T3B-T4B))/DEP(6,6)
END IF
C}
C COMPUTE THE HG TERM
C IF(DEP(1,1).LE.0.0) THEN
C TEMP1(Index) = TEMP1(Index)+DEP(1,1)*0.5*
    (Z1-TEMP1(Index))
C $ X1I = Z1-TEMP1(Index)
C X2I = (Z1**2)-(TEMP1(Index)**2)
C X3I = (Z1**3)-(TEMP1(Index)**3)
C X4I = (Z1**4)-(TEMP1(Index)**4)
C X5I = (Z1**5)-(TEMP1(Index)**5)
C T1 = TEMPC(Index)+TEMPC(Index)>XX1
C T2 = ((TEMP1(Index)**4)+DEP(1,1)*XX1)/4.0
C T3 = (IX6+DEP(1,1))/2.0
C T4 = (TEMP1(Index)**2)+II3+DEP(1,1))/6.0
C T5 = TEMPC(Index)+TEMPC(Index)**2+XX1
C T6 = TEMPC(Index)+XX3/3.0
C TEMP3(Index) = TEMP3(Index) + ((T1+DEP(1,1)*(T6-T6+T2+T3-T4))
    )/(DEP(1,1)*0.5))
C END IF
C ENDIF
C TEMPC(Index) = TEMPS(LYHUM,ISTGPH) + TEMP3B(1
C INDEX)
C C TRANSVERSE COMPONENTS IN THE Y'DIRECTION
C C
COMPUTE THE R2 TERM

\[
AR2(Index) = DEP(2,2)*(1/3.)*(Z2**3-TEBPB(Index)*/3) + AR2(Index) + AR2BC(Index)
\]

\[
AR2B(Index) = DEP(2,2)*(1/3.)*(Z2+AC0ST)**3-Z2**3
\]

COMPUTE THE G TERM, GEAR TERM AND ACCUMULATE

\[
TEMP(LYHUM,IHTGPH) = TEMPD(Index) + ((DEP(2,2)**3-TEBPB(Index)**3)/6.0 - (Z2-TEMPB(Index)**2/2.0)) / THICK + TEMP4(Index) + TEMP4B(Index)
\]

\[
TEMP4B(Index) = TEHP7(LYHUM,IHTGPH)+AC0ST/THICK - DEP(2,2)*(Z2+AC0ST)**3-Z2**3)/6.0 - AC0ST**Z2**2)/THICK
\]

COMPUTE THE HG TERM

IF (PROPER(26,MATRL) .NE. 0.0) THEN

\[
TEMP = TEMP2(Index) + DEP(2,2)*(Z2-TEMPB(Index)) + TEMP2B(Index)
\]

\[
YY1 = Z2-TEMPB(Index)
YY2 = Z2**2-TEMPB(Index)**2
YY3 = Z2**3-TEMPB(Index)**3
YY4 = Z2**4-TEMPB(Index)**4
YYB = Z2**B-TEMPB(Index)**B
T1 = TEMP2(Index)+TEMPB(Index)**Y1
T2 = (TEMP2(Index)**4+DEP(2,2)**YY1)/4.0
T3 = (YY1+DEP(2,2))/2.0
T4 = TEMP2(Index)**2+YY1*DEP(2,2)/6.0
T5 = TEMP2(Index)+TEMPB(Index)**Y1
T6 = TEMP2(Index)+YY3/3.0
TEMP4(Index) = TEMP2(Index) + (T1+DEP(2,2))**Y1/DEP(2,2) + TEMP4(Index)
\]

ELSE

\[
TEMP = TEMP2(Index) + DEP(2,2)*(Z2-TEMPB(Index))
\]

\[
YY1 = Z2-TEMPB(Index)
\]

\[
YY2 = Z2**2-TEMPB(Index)**2
YY3 = Z2**3-TEMPB(Index)**3
YY4 = Z2**4-TEMPB(Index)**4
YYB = Z2**B-TEMPB(Index)**B
T1 = TEMP2(Index)+TEMPB(Index)**Y1
T2 = (TEMP2(Index)**4+DEP(2,2)**YY1)/4.0
T3 = (YY1+DEP(2,2))/2.0
T4 = TEMP2(Index)**2+YY1*DEP(2,2)/6.0
T5 = TEMP2(Index)+YY3/3.0
TEMP4(Index) = TEMP2(Index) + (T1+DEP(2,2))**Y1/DEP(2,2) + TEMP4(Index)
\]

REPRODUCED WITH PERMISSION OF THE COPYRIGHT OWNER. FURTHER REPRODUCTION PROHIBITED WITHOUT PERMISSION.
C YXZ = \(2^{2}\) - \((\text{TMPD}(\text{INDEX})^{*2})\)
C YX2 = \(2^{2}\) - \((\text{TMPD}(\text{INDEX})^{*2})\)
C YX3 = \(2^{2}\) - \((\text{TMPD}(\text{INDEX})^{*2})\)
C YX4 = \(2^{2}\) - \((\text{TMPD}(\text{INDEX})^{*2})\)
C YX5 = \(2^{2}\) - \((\text{TMPD}(\text{INDEX})^{*2})\)
C T1 = TMPD(\text{INDEX})\times\text{TMPD}(\text{INDEX})\times\text{YXZ}
C T2 = ((\text{TMPD}(\text{INDEX})^{*2})\times\text{DEP}(2,2)\times\text{YXZ})/4.0
C T3 = (\text{YXZ}\times\text{DEP}(2,2))/20.0
C T4 = ((\text{TMPD}(\text{INDEX})^{*2})\times\text{YXZ}\times\text{DEP}(2,2))/6.0
C T6 = \text{TMPD}(\text{INDEX})\times\text{TMPD}(\text{INDEX})\times\text{YXZ}
C T6 = \text{TMPD}(\text{INDEX})\times\text{YXZ}\times\text{DEP}(2,2)
C \text{TEMP}(\text{INDEX}) = \text{TEMP}(\text{INDEX}) + ((\text{T1} + (\text{DEP}(2,2)\times(\text{T5}-\text{T6}\times\text{T2}\times\text{T4})/((\text{DEP}(2,2)\times0.8))))
C \text{END IF}
C \text{END IF}
C \text{TEMPD}(\text{INDEX}) = \text{TEHP}(\text{LYHUM},\text{IBTGP}) + \text{TEMP}(\text{INDEX})
C \text{TEMP}(\text{INDEX}) = \text{Z1} + \text{ACONST}
C \text{TEMPD}(\text{INDEX}) = \text{Z2} + \text{ACONST}
C \text{END IF}
C \text{CONTINUE}
C \text{WRITE} \ (\text{HIST})(\text{ELB}, \text{LYHUM}, \text{HIP}, ((\text{CRACK}(\text{LYHUM}, \text{LYHUM}), \text{K} = 1, 260), \text{L} = 1, \text{HIP})
C \text{WRITE} \ (\text{THBAR})(\text{ELB}, \text{LYHUM}, \text{HIP}, ((\text{TEB}(\text{LYHUM}, \text{LYHUM}, \text{K} = 1, 80), \text{L} = 1, \text{HIP}), ((\text{HIST}(\text{LYHUM}, \text{LYHUM}, \text{K} = 2, 1, 70), \text{L} = 1, \text{HIP}), (\text{PROPER}(\text{KH}, \text{MATRL}), \text{KH} = 1, 26))
C \text{CONTINUE}
C \text{FOR EACH ELEMENT GPBT VERTICAL LINE COMPUTE THE K1 AND K2 FACTOR}
C \text{FOR EACH GPBT IN A VERTICAL LINE COMPUTE THE F1 AND F2 FACTOR}
C \text{IF} (\text{IFLAG} .EQ. 4) \text{THEN}
C \text{IF} (\text{BLAYRS} .GT. 1) \text{THEN}
C \text{DO} \text{LY} = 1, \text{BLAYRS}
C \text{CALL} \text{LYIBFO}(\text{LY}, \text{LTHICK}, \text{ZS}, \text{MATRL}, \text{DCS}, \text{BBEL}, \text{ELB}, \text{LYHUM}, \text{HIPX}, \text{HIPETA}, \text{HIPSI})
C \text{IPB} = \text{HIPX} + \text{HIPETA} + \text{HIPSI}
C \text{IPB} = \text{HIPX} + \text{HIPETA}
C \text{IF} (\text{PROPER}(\text{KH}, \text{MATRL}) .NE. 0.0) \text{THEN}
C \text{DO} \text{IPB} = 1, \text{IPB}
C \text{IPOINT} = \text{IPB}/\text{II}
C \text{INDEX} = \text{IPB} - \text{IPOINT}\times\text{II}
C \text{IF} (\text{INDEX} .EQ. 0) \text{INDEX} = \text{II}
C \text{AK1(ELB,INDEX) = AK1(\text{INDEX}) + 2/((\text{TEMP}(\text{INDEX}) + \text{TEMP}(\text{INDEX}))}
C \text{AK2(ELB,INDEX) = AK2(\text{INDEX}) + 2/((\text{TEMP}(\text{INDEX}) + \text{TEMP}(\text{INDEX}))}
C \text{CALL GETTHK(IPB,ELB,LYHUM,THICKE,RAD)}
C \text{AF1(ELB,LY,IPB) = TEMPS(LY,IPB)/TEMP6(\text{INDEX})}
C \text{AF2(ELB,LY,IPB) = TEMPS(LY,IPB)/TEMP6(\text{INDEX})}
C \text{END DO}
C \text{END IF}
END DO
ENDIF
ENDIF
IF (CDIVER .EQ. 999.0) ICHECK = 1
200 CONTINUE
210 CONTINUE
C
ITEMP = IHISTY
IHISTY = IHIST1
IHIST1 = ITEMP
ITEMP = ITHSPG
ITHSPG = ITHSGL
ITHSGL = ITEMP
C
IF (ICHECK .EQ. 0) THEN
REWIND IHISTY
REWIND IHIST1
REWIND IHIST2
REWIND ITHSPG
REWIND ITHSGL
REWIND ITHSGL
C
ICP = 62
REWIND ICP
C
DO 220 ELHUM = 1, HELEM
CALL ELDINFO (ELSUM, ITYPE, ESREL, IFLAG, ISTART, LINES)
CALL ELISTM (ELSUM, IDENT, IBTCODE, HIPXI, HIPETA, HIPSI, IMAT, TECK)
1
DO 220 LBUM = 1, HLAYRS
CALL LLYHFD (LYSUM, LTHICK, ZS, MATRT, DCS, HHEL, ITHUM, HIPXI, HIPETA, HIPSI)
C
C RECOVER ARRAY 'HISTOCRACK' 'HISTOCRACK' FOR THE CURRENT ELEMENT.
C THESE ARE REQUIRED FOR THE UPDATE PROCEDURE IN THE CONVERGENCE
C CRITERION EMPLOYED. STORES ALL THE MATERIAL INFO ABOUT POINT.
C
HIP = HIPXI*HIPETA*HIPSI
READ (IHISTY) ELHUH, LYHUH, HGAUS, ((CRACK(L,K,LYHUM),K = 1,280), L = 1, HGAUS)
IF (ELHUH .NE. ELHUM .OR. LYHUH .NE. LYBUMB .OR. HGAUS .NE. HGAUS) THEN
WRITE (*, *) 'ERROR READING IHISTY IN CHKRC',
WRITE (*, *) 'ELHUM', ELHUH, 'ELHUH', ELHUH,
WRITE (*, *) 'LYHUM', LYHUM, 'LYBUMB', LYBUMB,
WRITE (*, *) 'HGAUS', HGAUS
STOP
ENDIF
C
READ (ITHSPG) ELHUH, LYHUH, HGAUS, ((HISTO(L,K,LYHUM),K = 1,280), L = 1, HGAUS),
((PROPER(KM,MATRT),KM = 1, 28))
IF (ELHUH .NE. ELHUM .OR. LYHUH .NE. LYBUMB .OR. HGAUS .NE. HGAUS) THEN
WRITE (*, *) 'ERROR READING ITHSPG IN CHKRC',

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
WRITE (*, *) 'ELNUM', ELNUM, 'ELNUM', ELNUM,
1 'LYNUM', LYNUM, 'LYNUM', LYNUM, 'NIP', NIP,
2 'SGAUS', SGAUS

STOP

ENDIF

C

THE UPDATED 'HISTO', 'CRACK' & 'TEHSTF' ARRAYS OF THE CURRENT
BLOCK OF ELEMENTS IS STORED

C

WRITE (IHIST2) ELNUM, LYNUM, NIP, ((CRACK(L,K,LYNUM), K
1 = 1, HIP), L = 1, NIP)

WRITE (ITHSO2) ELNUM, LYNUM, NIP, MATRL, ((TEHSTF(L,K,
1 LYNUM), K = 1, HIP), L = 1, NIP), (PROPER(250,K,MATRL), K = 1,
3)

WRITE (ICP) ELHUM, LYHUM, HIP, (AF1(ELHUM,LYHUM,KJ),
1 KJ = 1, HIP), (AF2(ELHUM,LYHUM,KJ), KJ = 1, HIP)

C

C INCLUDE(PROCESS)

SUBROUTINE CHKCH2(TOLER, IELEM, IOUT, PROPER, COHVER, HSTIFF, HHEL
$ , IHCREM, HIT, ITYPE ,L, TERROR, IFLAG)

IMPLICIT REAL*8(A-H,O-Z)

C...SWITCHES: REHUMB=100:10,FORMAT=900:10
C...SWITCHES:

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

C IDGEL PSTSTF OTDSEN CRKPTF TSSTIF CMPSFT

C SUBCON COB23 COB12 STATUS USRELD UDFMAT

C ROCRAK

C

C***************************************************************************

REAL*8 LTHICK
CHARACTER*100,DUMMY
COMMON/INPUT/EPSP,EPSET,EPSETI,NIP,IINTCOD
COMMON/BLOCKS/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHISTl
$ ,IHIST2
COMMON/ASCI/TEHSTF(27,80,18),ITRSO1,ITRSO2
COMMON/LAYERB/ITRICK(9),ZS(9),DCS(3,3),LAYERS,MATRL,LYNUM
COMMON/UTILI/STRESS(6),STRAIN(6),DUMMY
COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEVP,LDFAIL,LDEVU,LDEVST
COMMON/TRANS/V1(3),V2(3),V3(3)
COMMON/MATERI/DEP(6,6)
COMMON/ICRPKP/I2CEFO,ICCRM

DIMENSION PROPER(25),EPSILH(6),PVAL(3),PDIR(3,3),PSTDIR(3,3)

$ ,DTSTIF(6,6),TEMP11(6,6)

LOGICAL CHKCUS

C

C SUBROUTINE TO CHECK CONVERGENCE FOR EACH GAUSS POINT OF
C CONCRETE LAYER 'J' OF ELEMENT 'IELEM'. IF NOT CONVERGED FOR
C ANY GAUSS POINT THE MATERIAL PARAMETERS WILL BE MODIFIED

C

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
APPROPRIATELY AND THE NEW ELEMENT STIFFNESS WILL BE CALCULATED.

'CHKGUS' DETERMINES IF THE MATERIAL LAW AT THE CURRENT GAUSS POINT HAS CONVERGED.

CHKGUS = .FALSE. ..... CONVERGENCE

CHKGUS = .TRUE. ..... SHOULD UPDATE THE 'D' MATRIX

MAXCRK = 10
MAXCR2 = 30

CHKGUS = .FALSE.
IF H = PROPER(24)
IF C = PROPER(23)
EPSMAX = 0.0
EPSMIN = 0.0
EPSMID = 0.0
EPS1 = 0.0
EPS2 = 0.0
EPS3 = 0.0
EPSX = 0.0
EPSY = 0.0
EPSZ = 0.0
GAMAXY = 0.0
GAMAYZ = 0.0
GAMAXZ = 0.0
SIGMA1 = 0.0
SIGMA2 = 0.0
SIGMA3 = 0.0
ICOU = 0
IF LG1 = 0
IF LG2 = 0
IF LG3 = 0
ITCRK = 0
D11 = 0.0
D12 = 0.0
D13 = 0.0
D21 = 0.0
D22 = 0.0
D23 = 0.0
D31 = 0.0
D32 = 0.0
D33 = 0.0
PERCET = 0.0
J = LYNUM

CALL IOGET (LDEV1, 96, '(496)', 5)

EPSX = STRAIN(1)
EPSY = STRAIN(2)
EPSZ = STRAIN(3)
GAMAXY = STRAIN(4)
GAMAYZ = STRAIN(5)
GAMAXZ = STRAIN(6)
GHI = DMAX1(DMAX1(DMAX1(DMAX1(DMAX1(DABS(EPSX), DABS(EPSY)), DABS(EPSY)), DABS(EPSX)), DABS(EPSY)), DABS(EPSX), DABS(EPSZ), DABS(GAMAXY), DABS(GAMAYZ), DABS(GAMAXZ))
IF (DABS(EPSX) .LT. 1.0.E-3*GHI) EPSX = 0.0
IF (DABS(EPSY) .LT. 1.0.E-3*GHI) EPSY = 0.0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IF (DABS(EPSZ) .LT. 1.E-3*GHI) EPSZ = 0.0
IF (DABS(GAMAY) .LT. 1.E-3*GHI) GAMAY = 0.0
IF (DABS(GAMAY2) .LT. 1.E-3*GHI) GAMAY2 = 0.0
IF (DABS(GAMAZ) .LT. 1.E-3*GHI) GAMAZ = 0.0

CHAXCRK = MAX. ALLOWABLE CRACKS AT THE POINT

IF THERE EXIST CRACKS AT THIS POINT CHECK THE
CRACKED POINT'S STATUS; CALCULATE PRINCIPAL
STRAINS,'EPS1', 'EPS2' AND EPS3 FOR INTACT CONCRETE
AND DETERMINE PRINCIPAL STRESSES 'SIGMA1', 'SIGMA2' &
'SIGMA3'.

IF (CRACK(L,1,J) .GE. 1.) THEN

WITH THE GLOBAL STRAINS AT THE GAUSS POINT OF THE
CURRENT LAYER CALCULATE AND SAVE THE MAX. PRINC.
STRAIN DIRECTION,'PRIDIR', FOR THE CRACKED POINT.

EPSILH(1) = EPSX
EPSILH(2) = EPSY
EPSILH(3) = EPSZ
EPSILH(4) = GAMAXY
EPSILH(5) = GAMAZ
EPSILH(6) = GAMAYZ

IF IFG = 1
CALL PSTSTD(EPSILH, PVAL, PSTDIR, IFG, IDUT)

EPSMAX = PVAL(1)
EPSMID = PVAL(2)
EPSTMID = PVAL(3)
AEP = DMAX1(DMAX1(DABS(EPSMAX),DABS(EPSMID)),DABS(EPSMID))
IF (DABS(EPSMAX) .LT. 1E-3*AEP) EPSMAX = 0.0
IF (DABS(EPSMID) .LT. 1E-3*AEP) EPSMID = 0.0
IF (DABS(EPSMID) .LT. 1E-3*AEP) EPSMID = 0.0

C STRENGTH REDUCTION DUE TO CRACKING, AND RECOMPUTATION OF PARAMETERS
C FOR THE OTOSHI ENVELOPE, THE TOTAL STRAINS RECOVERED FOR UPDATING
C THE MATERIAL MODEL IF CONVERGENCE IS NOT OBSERVED POINTWISE.

FC = PROPER(8)
FT = PROPER(6)
IF (CRACK(L,1,J) .GE. 1.0) THEN
IF (IFC .EQ. 2) THEN
IF (EPSMAX .LT. PROPER(7)) THEN
FC = FC*(1.0-0.2*(EPSMAX/PROPER(7)))
ELSE
FC = 0.8*FC
ENDIF
ENDIF
FC = DMIN(HISTO(L,68,J),FC)
HISTO(L,68,J) = FC

CALL OTOSHI(FC, FT, PROPER, IDUT)
CALL CRKPT (EPSX, EPSY, EPSZ, GAHAXY, GAMAYZ, GAHAXZ, PROPER
1 , TOLEK, CHGKUS, RSTIFF, CONVER, IDUT, EPS1, EPS2, EPS3,
2 IELEM, MAIRC, MAICX2, SIGMA1, SIGMA2, SIGMA3, L, J, PDIR
3 , BLAYS, BEEL, ICTYPE, IFLAG, ISOFT, ITCRK)
C
AEP = DMAX1 (DKAX1 (DABS (EPS1), DABS (EPS2), DABS (EPS3)))
IF (DABS (EPS1) . LT. 1E-3 * AEP) EPS1 = 0.0
IF (DABS (EPS2) . LT. 1E-3 * AEP) EPS2 = 0.0
IF (DABS (EPS3) . LT. 1E-3 * AEP) EPS3 = 0.0
C
PRINCIPAL STRESS & DIRECTION OF INTACT CONCRETE ARE STORED.
C THIS WILL ALSO BE THE MAX. PRINCIPAL STRESS
C DIRECTION FOR THE CRACKED POINT.
C
SIGMA1 = DMIN1 (PROPER (6), SIGMA1)
SIGMA2 = DMIN1 (PROPER (6), SIGMA2)
SIGMA3 = DMIN1 (PROPER (6), SIGMA3)
IF (IFLAG . EQ. 4) THEN
DOT1 = V3 (1) * PDIR (1,1) + V3 (2) * PDIR (1,2) + V3 (3) * PDIR (1,3)
DOT2 = V3 (1) * PDIR (2,1) + V3 (2) * PDIR (2,2) + V3 (3) * PDIR (2,3)
DOT3 = V3 (1) * PDIR (3,1) + V3 (2) * PDIR (3,2) + V3 (3) * PDIR (3,3)
IF (DABS (DOT1) . GE. 0.99) SIGMA1 = 0.0
IF (DABS (DOT2) . GE. 0.99) SIGMA2 = 0.0
IF (DABS (DOT3) . GE. 0.99) SIGMA3 = 0.0
ENDIF
HISTO (L, 61, J) = SIGMA1
HISTO (L, 62, J) = SIGMA2
HISTO (L, 63, J) = SIGMA3
C
HISTO (L, 10, J) = PDIR (1,1)
HISTO (L, 11, J) = PDIR (1,2)
HISTO (L, 12, J) = PDIR (1,3)
HISTO (L, 13, J) = PDIR (2,1)
HISTO (L, 14, J) = PDIR (2,2)
HISTO (L, 15, J) = PDIR (2,3)
HISTO (L, 16, J) = PDIR (3,1)
HISTO (L, 17, J) = PDIR (3,2)
HISTO (L, 18, J) = PDIR (3,3)
C
THE NEW VERSION STIFFENING MODEL PARAMETERS AND THE COUPLING DATA
C ARE RECOMPUTED IF NECESSARY.
C
CALL TSSTIF (EPSX, EPSY, EPSZ, GAHAXY, GAMAYZ, GAHAXZ, SIGMA1
1 , SIGMA2, SIGMA3, DSTIF, L, J, CONVER, RSTIFF, CHGKUS,
2 PROPER, TOLEK, IELEM, BEEL, IFS, IFC, IFCoup, IFLAG1, IFLAG2
3 , IFLAG3, ISOFT, ITCRK, TEMPl1)
C
HISTO (L, 44, J) = PSTDIR (1,1)
HISTO (L, 45, J) = PSTDIR (1,2)
HISTO (L, 46, J) = PSTDIR (1,3)
HISTO (L, 47, J) = PSTDIR (2,1)
HISTO (L, 48, J) = PSTDIR (2,2)
HISTO (L, 49, J) = PSTDIR (2,3)
HISTO (L, 50, J) = PSTDIR (3,1)
HISTO (L, 51, J) = PSTDIR (3,2)
HISTO (L, 52, J) = PSTDIR (3,3)
GO TO 120
ENDIF
C
FC = PROPER (6)
FT = PROPER (6)
C WRITE(*,*) 'FT', FT

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CALL OTOSEB (FC, FT, PROPER, IOUT)

DETERMINE PRINCIPAL STRAINS AND DIRECTION COSINES OF STRAIN TENSOR, BEFORE CRACKING OCCURS AT THE POINT.

EPSILN(1) = EPSX
EPSILN(2) = EPSY
EPSILN(3) = EPSZ
EPSILN(4) = GAMAXY
EPSILN(5) = GAMAZY
EPSILN(6) = GAMAXZ

IF G = 1
CALL PSTSTN (EPSILH, PVAL, PDIR, IFG, IOUT)
EPS1 = PVAL(1)
EPS2 = PVAL(2)
EPS3 = PVAL(3)

AEP = DMAX1(DMAX1(DABS(EPS1),DABS(EPS2)),DABS(EPS3))
IF (DABS(EPS1) LT 1E-3*AEP) EPS1 = 0.0
IF (DABS(EPS2) LT 1E-3*AEP) EPS2 = 0.0
IF (DABS(EPS3) LT 1E-3*AEP) EPS3 = 0.0

IF (HISTO(L,29,J) EQ 4.) THEN

IF THE POINT IS CRUSHED SAVE THE CURRENT STRAINS ONLY.

HISTO(L,22,J) = EPS1
HISTO(L,23,J) = EPS2
HISTO(L,24,J) = EPS3
HISTO(L,10,J) = PDIR(1,1)
HISTO(L,11,J) = PDIR(1,2)
HISTO(L,12,J) = PDIR(1,3)
HISTO(L,13,J) = PDIR(2,1)
HISTO(L,14,J) = PDIR(2,2)
HISTO(L,15,J) = PDIR(2,3)
HISTO(L,16,J) = PDIR(3,1)
HISTO(L,17,J) = PDIR(3,2)
HISTO(L,18,J) = PDIR(3,3)

CHECK THE NEXT GAUSS POINT

RETURN
ENDIF

USING THE EXISTING CONSTITUTIVE MATRIX DETERMINE PRINCIPAL STRESSES; ASSUME THAT PRINCIPAL STRESS AND STRAIN DIRECTIONS COINCIDE (FOR UNCRACKED PORTION OF THE CONCRETE).

DO IM = 1, 6
   DO IN = 1, 6
      DEP(IM,IN) = 0.0
   END DO
END DO

RECOVER THE SOLID/ SHELL STIFFNESS MATRIX FOR THE ELEMENT AND COMPUTE THE PRINCIPAL STRESSES.
\[ D_{11} = \text{HISTO}(L,1,J) \]
\[ D_{12} = \text{HISTO}(L,2,J) \]
\[ D_{13} = \text{HISTO}(L,6,J) \]
\[ D_{21} = \text{HISTO}(L,2,J) \]
\[ D_{22} = \text{HISTO}(L,3,J) \]
\[ D_{23} = \text{HISTO}(L,6,J) \]
\[ D_{31} = \text{HISTO}(L,6,J) \]
\[ D_{32} = \text{HISTO}(L,6,J) \]
\[ D_{33} = \text{HISTO}(L,4,J) \]
\[ \text{SIGM}_{A1} = D_{11} \cdot \text{EPS1} + D_{12} \cdot \text{EPS2} + D_{13} \cdot \text{EPS3} \]
\[ \text{SIGM}_{A2} = D_{21} \cdot \text{EPS1} + D_{22} \cdot \text{EPS2} + D_{23} \cdot \text{EPS3} \]
\[ \text{SIGM}_{A3} = D_{31} \cdot \text{EPS1} + D_{32} \cdot \text{EPS2} + D_{33} \cdot \text{EPS3} \]
\[ \text{SIGM}_{A1} = \text{MAX1}(\text{PROPER}(6), \text{SIGM}_{A1}) \]
\[ \text{SIGM}_{A2} = \text{MAX1}(\text{PROPER}(6), \text{SIGM}_{A2}) \]
\[ \text{SIGM}_{A3} = \text{MAX1}(\text{PROPER}(6), \text{SIGM}_{A3}) \]

If (IFLAG .EQ. 4) THEN
\[ \text{DOT1} = V3(1) \cdot \text{PDIR}(1,1) + V3(2) \cdot \text{PDIR}(1,2) + V3(3) \cdot \text{PDIR}(1,3) \]
\[ \text{DOT2} = V3(1) \cdot \text{PDIR}(2,1) + V3(2) \cdot \text{PDIR}(2,2) + V3(3) \cdot \text{PDIR}(2,3) \]
\[ \text{DOT3} = V3(1) \cdot \text{PDIR}(3,1) + V3(2) \cdot \text{PDIR}(3,2) + V3(3) \cdot \text{PDIR}(3,3) \]
IF (DABS(DOT1) .GE. 0.99) \text{SIGM}_{A1} = 0.0
IF (DABS(DOT2) .GE. 0.99) \text{SIGM}_{A2} = 0.0
IF (DABS(DOT3) .GE. 0.99) \text{SIGM}_{A3} = 0.0
ENDIF

\[ \text{SABS} = \text{MAX1}(\text{MAX1}(\text{DABS}(\text{SIGM}_{A1}), \text{DABS}(\text{SIGM}_{A2})), \text{DABS}(\text{SIGM}_{A3})) \]
IF (DABS(\text{SIGM}_{A1}) .LT. \text{SABS} \cdot 1.E-4) \text{SIGM}_{A1} = 0.0
IF (DABS(\text{SIGM}_{A2}) .LT. \text{SABS} \cdot 1.E-4) \text{SIGM}_{A2} = 0.0
IF (DABS(\text{SIGM}_{A3}) .LT. \text{SABS} \cdot 1.E-4) \text{SIGM}_{A3} = 0.0

\[ \text{HISTO}(L,61,J) = \text{SIGM}_{A1} \]
\[ \text{HISTO}(L,62,J) = \text{SIGM}_{A2} \]
\[ \text{HISTO}(L,63,J) = \text{SIGM}_{A3} \]

120 CONTINUE
IFLAG1 = 0
IFLAG2 = 0
IFLAG3 = 0

C GO TO VECCHIO COLLABS MODEL FOR CRACKED CONCRETE
C
IF (IFC .EQ. 1) THEN
IF (\text{CRACK}(L,1,J) .GE. 1.0) THEN
IF (DABS(\text{SIGM}_{A3}) .GT. \text{PROPER}(S) \cdot 0.30 \ .AND. \ \text{SIGM}_{A3} .LT. 0.0)
1
THEN
CALL CMPSFT (\text{EPSMAX}, \text{EPSMIN}, \text{EPSMIN}, \text{SIGM}_{A1}, \text{SIGM}_{A2},
1
\text{SIGM}_{A3}, \text{PROPER}, \text{EPS1}, \text{EPS2}, \text{EPS3}, L, J, IOUT,
2
\text{COHER}, \text{BISTIFF}, \text{CHECUS}, \text{TOLER})
GOTO 130
ENDIF
ENDIF

C STORE THE TOTAL STRAINS FOR WRITING OUT LATER
C
IF (\text{CRACK}(L,1,J) .GE. 1.0) THEN
\text{HISTO}(L,38,J) = \text{EPSMAX}
\text{HISTO}(L,39,J) = \text{EPSMIN}
\text{HISTO}(L,40,J) = \text{EPSMIN}
\text{HISTO}(L,41,J) = \text{EPSMAX}
\text{HISTO}(L,42,J) = \text{EPSMIN}
HISTO(L,43,J) = EPSMIN
ENDIF

C C C STRAINING IN THE OPPOSITE DIRECTION
C C C
IF (EPS1*HISTO(L,31,J) .LT. 0.0) IFLAG1 = 1
IF (EPS2*HISTO(L,32,J) .LT. 0.0) IFLAG2 = 1
IF (EPS3*HISTO(L,33,J) .LT. 0.0) IFLAG3 = 1
IF (IFLAG1.EQ.1 .OR. IFLAG2.EQ.1 .OR. IFLAG3.EQ.1) THEN
CALL SUBCON (PDIR, SIGMA1, SIGMA2, SIGMA3, PROPER, L, J, 1
       CONVER, IFLAG1, IFLAG2, IFLAG3, EPS1, EPS2, EPS3, IFC)
IF (CRAK(L,1,J) .GE. 1.0) GO TO 130
RETURN
ENDIF

2 )) THEN
    IF (CRAK(L,1,J) .GE. 1.0) THEN
      HISTO(L,10,J) = PDIR(1,1)
      HISTO(L,11,J) = PDIR(1,2)
      HISTO(L,12,J) = PDIR(1,3)
      HISTO(L,13,J) = PDIR(2,1)
      HISTO(L,14,J) = PDIR(2,2)
      HISTO(L,15,J) = PDIR(2,3)
      HISTO(L,16,J) = PDIR(3,1)
      HISTO(L,17,J) = PDIR(3,2)
      HISTO(L,18,J) = PDIR(3,3)
    ENDIF
C C C IF THE POINT WAS UNLOADING OR RELOADING IN  C C C THE LAST ITERATION ....
C C C
C
C IF (HISTO(L,30,J).EQ.2 .OR. HISTO(L,30,J).EQ.3.) THEN
C CALL SUBCON (PDIR, SIGMA1, SIGMA2, SIGMA3, PROPER, L, J, 1
C       CONVER, IFLAG1, IFLAG2, IFLAG3, EPS1, EPS2, EPS3, IFC)
C IF (CRAK(L,1,J) .GE. 1.0) GO TO 130
C RETURN
CENDIF
IF (HISTO(L,30,J).EQ.1.) THEN
   *----------------------------------------------------------*
   * L O A D I N G *                                           *
   *----------------------------------------------------------*
   THE POINT WAS PREVIOUSLY LOADING (INCREASE IN STRAINS)
   AND CONTINUES TO DO SO.
C C C SAVE INTACT CONCRETE'S CURRENT PRINCIPAL STRAINS
   'EPS1' & 'EPS2' AT THIS POINT.
C C C
HISTO(L,22,J) = EPS1
HISTO(L,23,J) = EPS2
HISTO(L,24,J) = EPS3
IF (HISTO(L,29,J) .EQ. 2.) THEN
    

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
COMPARE THE CURRENT PRINCIPAL STRESSES WITH THOSE AT ULTIMATE. IF THERE IS OVERTAKING THEN THE MATERIAL IS IN THE POST-ULTIMATE REGION.

SIGMA1 = HISTO(L,34,J)
SIGMA2 = HISTO(L,35,J)
SIGMA3 = HISTO(L,36,J)
STATE = HISTO(L,28,J)
IF (STATE .EQ. (-100.)) THEN
  IF (DABS(SIGMA1) .GT. DABS(SIGMA3)) THEN
    HISTO(L,10,J) = PDIR(1,1)
    HISTO(L,11,J) = PDIR(1,2)
    HISTO(L,12,J) = PDIR(1,3)
    HISTO(L,13,J) = PDIR(2,1)
    HISTO(L,14,J) = PDIR(2,2)
    HISTO(L,15,J) = PDIR(2,3)
    HISTO(L,16,J) = PDIR(3,1)
    HISTO(L,17,J) = PDIR(3,2)
    HISTO(L,18,J) = PDIR(3,3)
    CALL COU23 (L, J, CONVER, BSTIFF, SIGMA1, SIGMA2, SIGMA3, PROPER, IOUT, TOLER, CHERUS, TELEM, IPC)
  GO TO 130
ENDIF

IF (STATE .EQ. (-110.) .OR. STATE .EQ. (-101) .OR. STATE .EQ. (-111.) ) THEN
  IF (DABS(SIGMA1) .GT. DABS(SIGMA1) .OR. DABS(SIGMA2) .GT. DABS(SIGMA3) .OR. DABS(SIGMA1) .GT. DABS(SIGMA2)) THEN
    HISTO(L,10,J) = PDIR(1,1)
    HISTO(L,11,J) = PDIR(1,2)
    HISTO(L,12,J) = PDIR(1,3)
    HISTO(L,13,J) = PDIR(2,1)
    HISTO(L,14,J) = PDIR(2,2)
    HISTO(L,15,J) = PDIR(2,3)
    HISTO(L,16,J) = PDIR(3,1)
    HISTO(L,17,J) = PDIR(3,2)
    HISTO(L,18,J) = PDIR(3,3)
    CALL COU23 (L, J, CONVER, BSTIFF, SIGMA1, SIGMA2, SIGMA3, PROPER, IOUT, TOLER, CHERUS, TELEM, IPC)
  GO TO 130
ENDIF

CHECK TO SEE IF CONVERGENCE IS ACHIEVED AT THE VICINITY OF THE ULTIMATE POINT.

IF (DABS(EPS1) .GE. .999 .AND. DABS(EPS1) .LE. 1.001) THEN
  IF (DABS(EPS2) .LE. .999 .AND. DABS(EPS2) .LE. 1.001) THEN
    IF (DABS(EPS3) .LE. .999 .AND. DABS(EPS3) .LE. 1.001) THEN
      THE STRAINS ARE NOT CHANGING. THEREFORE, THE LOADING IS
    ELSE
      THE MATERIAL IS IN THE POST-ULTIMATE REGION.
    ENDIF
  ELSE
    THE MATERIAL IS IN THE POST-ULTIMATE REGION.
  ENDIF
ENDIF

END
SUCH THAT THE POINT CONVERGES AT ITS ULTIMATE VALUE.

\[
\begin{align*}
\text{HISTO}(L,10,J) &= \text{PDIR}(1,1) \\
\text{HISTO}(L,11,J) &= \text{PDIR}(1,2) \\
\text{HISTO}(L,12,J) &= \text{PDIR}(1,3) \\
\text{HISTO}(L,13,J) &= \text{PDIR}(2,1) \\
\text{HISTO}(L,14,J) &= \text{PDIR}(2,2) \\
\text{HISTO}(L,15,J) &= \text{PDIR}(2,3) \\
\text{HISTO}(L,16,J) &= \text{PDIR}(3,1) \\
\text{HISTO}(L,17,J) &= \text{PDIR}(3,2) \\
\text{HISTO}(L,18,J) &= \text{PDIR}(3,3)
\end{align*}
\]

GO TO 130

ENDIF

ENDIF

IF THERE ARE CRACKS AT THIS POINT THEN THE UNLOADING STRAINS ARE THOSE CALCULATED AT THE ONSET OF CRACKING.

\[
\begin{align*}
\text{HISTO}(L,31,J) &= \text{EPS1} \\
\text{HISTO}(L,32,J) &= \text{EPS2} \\
\text{HISTO}(L,33,J) &= \text{EPS3}
\end{align*}
\]

IF THE STRESS POINT WAS IN THE PRE-FAILURE PHASE...

\[
\begin{align*}
\text{CALL CON23} (L, J, \text{CONVER}, \text{N STIFF}, \text{SIGMA1}, \text{SIGMA2}, \\
\text{SIGMA3, PROPER, JOUT, TOLER, CHKOUT, IELEM,} \\
\text{EPS1, EPS2, EPS3, INCREM, IT, PROPER, IELEM, ISEL, ITOUT,} \\
\text{I PVER, BSTIF, TOLE, PDIR, CHKOUT, ITYPE, DSTIF,} \\
\text{ITFC, IERROR, IFLG1, IFLG2, IFLG3, ICTCRK, ICPUP,} \\
\text{IFLAG)
\end{align*}
\]

GO TO 130

ENDIF

CONSIDER POSSIBILITY OF UNLOADING IN NEXT STEPS. THEREFORE, SAVE THE STRAINS FROM WHICH UNLOADING WILL START.

\[
\begin{align*}
\text{HISTO}(L,31,J) &= \text{EPS1} \\
\text{HISTO}(L,32,J) &= \text{EPS2} \\
\text{HISTO}(L,33,J) &= \text{EPS3}
\end{align*}
\]

IF THE STRESS POINT WAS IN THE PRE-FAILURE PHASE...

\[
\begin{align*}
\text{CALL CON23} (L, J, \text{CONVER}, \text{N STIFF}, \text{SIGMA1}, \text{SIGMA2}, \text{SIGMA3, L, J, EPS1, EPS2,} \\
\text{EPS3, INCREM, IT, PROPER, IELEM, ISEL, ITOUT,} \\
\text{IPVER, BSTIF, TOLE, PDIR, CHKOUT, ITYPE, DSTIF,} \\
\text{ITFC, IERROR, IFLG1, IFLG2, IFLG3, ICTCRK, ICPUP,} \\
\text{IFLAG)
\end{align*}
\]

GO TO 130

ENDIF

IF THE STRESS POINT, IN PREVIOUS ITERATION, WAS ON THE FAILURE ENVELOPE OR OUTSIDE THE ENVELOPE .........
   OR. HISTO(L,29,J).EQ.3.) THEN
  HISTO(L,10,J) = PDIR(1,1)
  HISTO(L,11,J) = PDIR(1,2)
  HISTO(L,12,J) = PDIR(1,3)
  HISTO(L,13,J) = PDIR(2,1)
  HISTO(L,14,J) = PDIR(2,2)
  HISTO(L,15,J) = PDIR(2,3)
  HISTO(L,16,J) = PDIR(3,1)
  HISTO(L,17,J) = PDIR(3,2)
  HISTO(L,18,J) = PDIR(3,3)
  CALL CON23 (L, J, CONVER, BSTIFF, SIGMA1, SIGMA2,
             SIGMA3, PROPER, IOUT, TOLER, CHK, KELEMS, IFC)
ENDIF
ENDIF
ENDIF

COMPARE THE CURRENT PRINCIPAL STRAINS, EPS1, EPS2 & EPS3.
WITH THE PREVIOUS VALUES SAVED IN 'HISTO' TO CHECK
UNLOADING OR RELOADING.

-----------------------------------------------

* UNLOADING OR RELOADING *

-----------------------------------------------

IF (DABS(EPS1).LT.999*DABS(HISTO(L,31,J)) AND DABS(EPS2).LT.
   999*DABS(HISTO(L,32,J)) AND DABS(EPS3).LT.999*DABS(HISTO(L
   2 ,33,J))) THEN
  IF (CRACK(L,1,J).EQ.0.0) THEN
    HISTO(L,10,J) = PDIR(1,1)
    HISTO(L,11,J) = PDIR(1,2)
    HISTO(L,12,J) = PDIR(1,3)
    HISTO(L,13,J) = PDIR(2,1)
    HISTO(L,14,J) = PDIR(2,2)
    HISTO(L,15,J) = PDIR(2,3)
    HISTO(L,16,J) = PDIR(3,1)
    HISTO(L,17,J) = PDIR(3,2)
    HISTO(L,18,J) = PDIR(3,3)
ENDIF

DETERMINE STATE OF STRESS AND RATIO OF PRINC. STRESSES
CALL STATUS (SIGMA1, SIGMA2, SIGMA3, STATE, J, PROPER, L, IFC)
HISTO(L,35,J) = STATE

IF THERE'S OVERSHOOTING IN THE ULTIMATE AND POST-
ULTIMATE REGION THEN THE DETECTED UNLOADING-RELOADING
CASE MUST BE TREATED WITH CAUTION IN ORDER TO GET
BACK TO STRESS-STRAIN CURVE THE STRAINS CAN TAKE A
PATH THAT CAN ACTUALLY BE DETECTED AS UNLOADING WITH
NO APPARENT NEED FOR CHANGE IN MATERIAL MODULUS.
ELIMINATE THE OVERSHOOTING PROBLEM FIRST AND THEN
COMPARE STRAINS.
   SIGM1F = HISTO(L,34,J)
   SIGM2F = HISTO(L,36,J)
   SIGM3F = HISTO(L,36,J)
   STATE = HISTO(L,28,J)
   IF (STATE.EQ.(-100.)) THEN
      IF (DABS(SIGMA3).GT.DABS(SIGM3F)) THEN
         CALL COH23 (L, J, CONVER, HSTIFF, SIGMA1, SIGMA2,
                     SIGMA3, PROPER, IOUT, TOLER, CHKGS, IELEM,
                     IF C)
         GO TO 130
      ENDIF
   ENDIF
   IF (STATE.EQ.(-110.) .OR. STATE.EQ.(-101.) .OR. STATE.EQ.(  
   -111.)) THEN
      IF (DABS(SIGMA1).GT.DABS(SIGM1F) .OR. DABS(SIGMA2).GT.  
          DABS(SIGMA2F) .OR. DABS(SIGMA3).GT.DABS(SIGM3F))  
          THEN
         CALL COH23 (L, J, CONVER, HSTIFF, SIGMA1, SIGMA2,
                     SIGMA1, PROPER, IOUT, TOLER, CHKGS, IELEM,
                     IF C)
         GO TO 130
      ENDIF
   ENDIF
ENDIF

IF (STATE.EQ.100. .OR. STATE.EQ.110. .OR. STATE.EQ.111.  
1 .OR. STATE.EQ.101) HISTO(L,29,J) = 1.
ENDIF
CALL UNRELD(EPS1,EPS2,EPS3,PROPER,CHKGS,HSTIFF,CONVER,L,J)
ENDIF

CALCULATE THE CURRENT STRAIN IN THE DIRECTION OF REBARS
130 CONTINUE
IF (CHKGS) THEN
THE MATERIAL LAW AT THE CURRENT GAUSS POINT HAS NOT
CONVERGED.UPDATE THE CONSTITUTIVE MATRIX.

CALL UPDMAT (MAXCRK, IOUT, PROPER, MAXCK2, L, J, DTSTIF, IDCUP
1 , IFC, IFLO1, IFLO2, IFLO3, ITCRK, TEMPI1, EPSX, EPSY,
2 EPSZ, GAMAXY, GAMAYZ, GAMAXZ, IELEM)
RETURN
ENDIF
IF (CRACK(L,1,J).GE.1 .AND. EPS1.GT.0.0) THEN
CHECK THE POSSIBILITY OF A NEW CRACK FORMATION THIS
CAN HAPPEN WHEN THE MAX. PRINCIPAL STRAIN OF INTACT
CONCRETE EXCEEDS ITS CORRESPONDING VALUE AT THE
INITIATION OF THE FIRST CRACK. SUCH CRACKS WERE NOT
CALL ROCKR (EPS1, SIGMA1, SIGMA2, SIGMA3, PROPER, L, J, IELEM
          , MEL, CRACK, NSTIFF, CONVER, IOUT, ITYPE, DNSTIF,
          IERROR, IFLAG1, IFLAG2, IFLAG3, ICTRK, ICoup, IFLAG)
        IF (CRACK) GO TO 130
      ENDIF
      RETURN
      END

C**INCLUDE(PROCESS)
SUBROUTINE GETSTF(ELNUM, IWTSPR, ILAYER, IOUT, ELAYRS, ICOUNT)
  IMPLICIT REAL*8(A-H,0-Z)
C...SWITCHES: REHUMB=100:10,FORMAT=900:10
C...SWITCHES:
C******************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C******************************************************************************
INTEGER ELNUM
COMMON/BLOCK6/HISTD(27,70,16),CRACK(27,250,16),HISTY,IBIST1
$ ,HIST2
COMMON/LPS/R/P/PROP(25,16)
COMMON/ABC/TESTF(27,80,16),ITSPG1,ITSPG2,ITSPG3
COMMON/LAYERA/ELLYFO(320,5000)
DIMENSION ICOUNT(27,16)

C PREPROCESSING TO DETERMINE LAYERS OF CONCRETE THAT ARE
C TENSION-STIFFENING I.E. CONCRETE LAYERS WITH STEEL ADJACENT TO IT.
C
IF (ICOUNT(IWTSPR,ILAYER) .LE. 2) THEN
  IF (ILAYER = 1 .LE. IOUT .AND. ILAYER.GE.1) THEN
    ICON = (ILAYER-1)*21 + 1
    MAC = ELLYFO(ICON,ELNUM)
    ES = PROPER(1,MAC)
    ICHT1 = ILAYER+21 + 1
    MATRL1 = ELLYFO(ICH1,ELNUM)
    ICK1 = PROPER(28,MATRL1)
    IF (ICK1 .EQ. 0) THEN
      EPSY1 = PROPER(5,MATRL1)/PROPER(1,MATRL1)
      SIGYD1 = PROPER(6,MATRL1)
      ARAT01 = PROPER(5,MATRL1)/ES
      ILY1 = ILAYER + 1
      IF (ILAYER .LT. 0 .OR. 0) THEN
        ICHT2 = (ILAYER-2)*21 + 1
        MATRL2 = ELLYFO(ICH2,ELNUM)
        ICK2 = PROPER(25,MATRL2)
        IF (ICK2 .EQ. 0) THEN
          EPSY2 = PROPER(6,MATRL2)/PROPER(1,MATRL2)
          SIGYD2 = PROPER(6,MATRL2)
          ARAT02 = PROPER(1,MATRL2)/ES
          ILY2 = ILAYER - 1
          ICHT3 = ICHT1 + 9
          ICHT4 = ICHT2 + 9
          AL1 = ELLYFO(ICH3,ELNUM)
          AM1 = ELLYFO(ICH3+1,ELNUM)
          AL2 = ELLYFO(ICH4,ELNUM)
          AM2 = ELLYFO(ICH4+1,ELNUM)
          AL3 = ELLYFO(ICH4+2,ELNUM)
  ENDIF
483

\[ \text{DOT} = a_1 \cdot a_2 + a_3 \cdot a_4 + a_5 \cdot a_6 \]

\[
\begin{align*}
\text{IF} \ (\text{DABS(DOT)} \geq 0.999 \ \text{AND} \ \text{DABS(DOT)} \leq 1.001) \ \text{THEN} \\
\text{EPSY} & = \text{EPSY}2 \\
\text{ILY} & = \text{ILY}2 \\
\text{SIGYLD} & = \text{SIGYD2} \\
\text{ARATIO} & = \text{ARAT02} \\
\text{AL1} & = \text{ELLYFO} (\text{ICHT4}, \text{ELHUM}) \\
\text{AM1} & = \text{ELLYFO} (\text{ICHT4}+1, \text{ELHUM}) \\
\text{AH1} & = \text{ELLYFO} (\text{ICHT4}+2, \text{ELHUM}) \\
\text{AL2} & = \text{ELLYFO} (\text{ICHT4}+3, \text{ELHUM}) \\
\text{AM2} & = \text{ELLYFO} (\text{ICHT4}+4, \text{ELHUM}) \\
\text{AH2} & = \text{ELLYFO} (\text{ICHT4}+5, \text{ELHUM}) \\
\text{AL3} & = \text{ELLYFO} (\text{ICHT4}+6, \text{ELHUM}) \\
\text{AM3} & = \text{ELLYFO} (\text{ICHT4}+7, \text{ELHUM}) \\
\text{AH3} & = \text{ELLYFO} (\text{ICHT4}+8, \text{ELHUM}) \\
\end{align*}
\]

\[
\begin{align*}
\text{ELSE} \\
\text{EPSY} & = \text{EPSY}1 \\
\text{ILY} & = \text{ILY}1 \\
\text{SIGYLD} & = \text{SIGYD1} \\
\text{ARATIO} & = \text{ARAT01} \\
\text{AL1} & = \text{ELLYFO} (\text{ICHT3}, \text{ELHUM}) \\
\text{AM1} & = \text{ELLYFO} (\text{ICHT3}+1, \text{ELHUM}) \\
\text{AH1} & = \text{ELLYFO} (\text{ICHT3}+2, \text{ELHUM}) \\
\text{AL2} & = \text{ELLYFO} (\text{ICHT3}+3, \text{ELHUM}) \\
\text{AM2} & = \text{ELLYFO} (\text{ICHT3}+4, \text{ELHUM}) \\
\text{AH2} & = \text{ELLYFO} (\text{ICHT3}+5, \text{ELHUM}) \\
\text{AL3} & = \text{ELLYFO} (\text{ICHT3}+6, \text{ELHUM}) \\
\text{AM3} & = \text{ELLYFO} (\text{ICHT3}+7, \text{ELHUM}) \\
\text{AH3} & = \text{ELLYFO} (\text{ICHT3}+8, \text{ELHUM}) \\
\end{align*}
\]

\[
\begin{align*}
\text{ENDIF} \\
\text{ICOUNT} (\text{ISTGPH}, \text{ILAYER}) & = \text{ICOUNT} (\text{ISTGPH}, \text{ILAYER}) + 1 \\
\text{IPHT} & = \text{ICOUNT} (\text{ISTGPH}, \text{ILAYER}) \\
\text{TEHSTF} (\text{ISTGPH}, 3 \cdot \text{IPHT}, \text{ILAYER}) & = \text{IPHT} \\
\text{TEHSTF} (\text{ISTGPH}, 12 \cdot \text{IPHT}, \text{ILAYER}) & = \text{EPSY} \\
\text{TEHSTF} (\text{ISTGPH}, 63 \cdot \text{IPHT}, \text{ILAYER}) & = \text{SIGYLD} \\
\text{TEHSTF} (\text{ISTGPH}, 66 \cdot \text{IPHT}, \text{ILAYER}) & = \text{ILY} \\
\text{INT} & = 9 \cdot (\text{IPHT} - 1) + 1 \\
\text{TEHSTF} (\text{ISTGPH}, 30 \cdot \text{INT}, \text{ILAYER}) & = \text{AL1} \\
\text{TEHSTF} (\text{ISTGPH}, 31 \cdot \text{INT}, \text{ILAYER}) & = \text{AM1} \\
\text{TEHSTF} (\text{ISTGPH}, 32 \cdot \text{INT}, \text{ILAYER}) & = \text{AH1} \\
\text{TEHSTF} (\text{ISTGPH}, 33 \cdot \text{INT}, \text{ILAYER}) & = \text{AL2} \\
\text{TEHSTF} (\text{ISTGPH}, 34 \cdot \text{INT}, \text{ILAYER}) & = \text{AM2} \\
\text{TEHSTF} (\text{ISTGPH}, 35 \cdot \text{INT}, \text{ILAYER}) & = \text{AH2} \\
\text{TEHSTF} (\text{ISTGPH}, 36 \cdot \text{INT}, \text{ILAYER}) & = \text{AL3} \\
\text{TEHSTF} (\text{ISTGPH}, 37 \cdot \text{INT}, \text{ILAYER}) & = \text{AM3} \\
\text{TEHSTF} (\text{ISTGPH}, 38 \cdot \text{INT}, \text{ILAYER}) & = \text{AH3} \\
\end{align*}
\]

\[
\begin{align*}
\text{ELSE IF} \ (\text{ILAYER} - 2 \cdot \text{GT} . \text{O}) \ \text{THEN} \\
\text{ICHT5} & = (\text{ILAYER} - 3) \cdot 21 + 1 \\
\text{MATRL3} & = \text{ELLYFO} (\text{ICHT5}, \text{ELHUM}) \\
\text{ICK3} & = \text{PROPER}(\text{ICHT5}, \text{MATRL3}) \\
\text{IF} \ (\text{ICK3} = \text{PROPER}(1, \text{MATRL3})) \ \text{THEN} \\
\text{EPSY} & = \text{PROPER}(\text{ICHT5}, \text{MATRL3})/\text{PROPER}(1, \text{MATRL3}) \\
\text{SIGYD3} & = \text{PROPER}(\text{ICHT5}, \text{MATRL3})/\text{PROPER}(1, \text{MATRL3}) \\
\text{ARATOS} & = \text{PROPER}(\text{ICHT5}, \text{MATRL3})/\text{PROPER}(1, \text{MATRL3}) \\
\text{ILY} & = \text{ILAYER} - 2 \\
\text{ICHT3} & = \text{ICHT1} + 9 \\
\text{ICHT6} & = \text{ICHT5} + 9 \\
\text{AL1} & = \text{ELLYFO} (\text{ICHT3}, \text{ELHUM}) \\
\end{align*}
\]

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
AMI = ELLYFO(ICHT3+1,ELNUM)
AH1 = ELLYFO(ICHT3+2,ELNUM)
AL2 = ELLYFO(ICHT6,ELNUM)
AM2 = ELLYFO(ICHT6+1,ELNUM)
AH2 = ELLYFO(ICHT6+2,ELNUM)
DOT = AL1*AL2 + AM1*AM2 + AH1*AH2
IF (DABS(DOT).GE.0.999 .AND. DABS(DOT).LE.1.001) THEN
IF (EPSY3 .GT. EPSY1) THEN
EPSY = EPSY3
ILY = ILY3
SIGYLD = SIGYD3
ARATIO = ARATIO3
AL1 = ELLYFO(ICHT6,ELNUM)
AM1 = ELLYFO(ICHT6+1,ELNUM)
AL2 = ELLYFO(ICHT6+3,ELNUM)
AM2 = ELLYFO(ICHT6+4,ELNUM)
AH2 = ELLYFO(ICHT6+5,ELNUM)
AL3 = ELLYFO(ICHT6+6,ELNUM)
AM3 = ELLYFO(ICHT6+7,ELNUM)
AH3 = ELLYFO(ICHT6+8,ELNUM)
ELSE
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARATIO1
AL1 = ELLYFO(ICHT3,ELNUM)
AM1 = ELLYFO(ICHT3+1,ELNUM)
AL2 = ELLYFO(ICHT3+3,ELNUM)
AM2 = ELLYFO(ICHT3+4,ELNUM)
AH2 = ELLYFO(ICHT3+5,ELNUM)
AL3 = ELLYFO(ICHT3+6,ELNUM)
AM3 = ELLYFO(ICHT3+7,ELNUM)
AH3 = ELLYFO(ICHT3+8,ELNUM)
ENDIF

ICOUNT(IPHTP,ILAYER) = ICOUNT(IPHTP,ILAYER) + 1
IPHT = ICOUNT(IPHTP,ILAYER)
TESTF(IPHTP,12+IPHT,ILAYER) = EPSY
TESTF(IPHTP,66+IPHT,ILAYER) = SIGYLD
TESTF(IPHTP,66+IPHT,ILAYER) = ILY
TESTF(IPHTP,66+IPHT,ILAYER) = ARATIO
IMT = 9*(IPHT-1) + 1
TESTF(IPHTP,30+INT,ILAYER) = AL1
TESTF(IPHTP,37+INT,ILAYER) = AM1
TESTF(IPHTP,30+INT,ILAYER) = AL2
TESTF(IPHTP,30+INT,ILAYER) = AM2
TESTF(IPHTP,30+INT,ILAYER) = AL3
TESTF(IPHTP,30+INT,ILAYER) = AM3
ELSE
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARATIO1
AL1 = ELLYFO(ICHT3,ELNUM)
AM1 = ELLYFO(ICHT3+1,ELNUM)
AH1 = ELLYFO(ICHT3+2,ELNUM)
AL2 = ELLYFO(ICHT3+3,ELNUM)
AM2 = ELLYFO(ICHT3+4,ELNUM)
AL3 = ELLYFO(ICHT3+5,ELNUM)
AM3 = ELLYFO(ICHT3+6,ELNUM)
AM3 = ELLYFO(ICHT3+7,ELNUM)
AM3 = ELLYFO(ICHT3+8,ELNUM)
ICOUNT(ISTOP3,ILAYER) = ICOUNT(ISTOP3,ILAYER) + 1
IPHT = ICOUNT(ISTOP3,ILAYER)
TEHSTF(ISTOP3,3*IPHT,ILAYER) = IPHT
TEHSTF(ISTOP3,12*IPHT,ILAYER) = EPSY
TEHSTF(ISTOP3,63*IPHT,ILAYER) = SIGYLD
TEHSTF(ISTOP3,66*IPHT,ILAYER) = ILY
TEHSTF(ISTOP3,60*IPHT,ILAYER) = ARATIO
INT = 9*(IPHT-1) + 1
TEHSTF(ISTOP3,30*INT,ILAYER) = AL1
TEHSTF(ISTOP3,31*INT,ILAYER) = AM1
TEHSTF(ISTOP3,32*INT,ILAYER) = AL2
TEHSTF(ISTOP3,33*INT,ILAYER) = AM2
TEHSTF(ISTOP3,34*INT,ILAYER) = AL3
TEHSTF(ISTOP3,35*INT,ILAYER) = AM3
TEHSTF(ISTOP3,36*INT,ILAYER) = AL4
TEHSTF(ISTOP3,37*INT,ILAYER) = AM4
TEHSTF(ISTOP3,38*INT,ILAYER) = AL5
TEHSTF(ISTOP3,39*INT,ILAYER) = AM5
ICOUNT(ISTOP3,ILAYER) = ICOUNT(ISTOP3,ILAYER) + 1
IPHT = ICOUNT(ISTOP3,ILAYER)
TEHSTF(ISTOP3,3*IPHT,ILAYER) = IPHT
TEHSTF(ISTOP3,12*IPHT,ILAYER) = EPSY
TEHSTF(ISTOP3,63*IPHT,ILAYER) = ARATIO
TEHSTF(ISTOP3,60*IPHT,ILAYER) = SIGYLD
TEHSTF(ISTOP3,66*IPHT,ILAYER) = ILY
INT = 9*(IPHT-1) + 1
TEHSTF(ISTOP3,30*INT,ILAYER) = AL1
TEHSTF(ISTOP3,31*INT,ILAYER) = AM1
TEHSTF(ISTOP3,32*INT,ILAYER) = AL2
TEHSTF(ISTOP3,33*INT,ILAYER) = AM2
TEHSTF(ISTOP3,34*INT,ILAYER) = AL3
TEHSTF(ISTOP3,35*INT,ILAYER) = AM3
TEHSTF(ISTOP3,36*INT,ILAYER) = AL4
TEHSTF(ISTOP3,37*INT,ILAYER) = AM4
TEHSTF(ISTOP3,38*INT,ILAYER) = AL5
ICOUNT(ISTOP3,ILAYER) = ICOUNT(ISTOP3,ILAYER) + 1
IPHT = ICOUNT(ISTOP3,ILAYER)
TEHSTF(ISTOP3,3*IPHT,ILAYER) = IPHT
TEHSTF(ISTOP3,12*IPHT,ILAYER) = EPSY
TEHSTF(ISTOP3,63*IPHT,ILAYER) = ARATIO
TEHSTF(ISTOP3,60*IPHT,ILAYER) = SIGYLD
TEHSTF(ISTOP3,66*IPHT,ILAYER) = ILY
INT = 9*(IPHT-1) + 1
TEHSTF(ISTOP3,30*INT,ILAYER) = AL1
TEHSTF(ISTOP3,31*INT,ILAYER) = AM1
TEHSTF(ISTOP3,32*INT,ILAYER) = AL2
TEHSTF(ISTOP3,33*INT,ILAYER) = AM2
TEHSTF(ISTOP3,34*INT,ILAYER) = AL3
TEHSTF(ISTOP3,35*INT,ILAYER) = AM3
TEHSTF(ISTOP3,36*INT,ILAYER) = AL4
TEHSTF(ISTOP3,37*INT,ILAYER) = AM4
TEHSTF(ISTOP3,38*INT,ILAYER) = AL5
ENDIF
ELSE
ICHT3 = ICHT1 + 9
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYLD1
ARATIO = ARATIO1
AL1 = ELLYFO(ICHT3,ELNUM)
AM1 = ELLYFO(ICHT3+1,ELNUM)
AL2 = ELLYFO(ICHT3+2,ELNUM)
AM2 = ELLYFO(ICHT3+3,ELNUM)
AL3 = ELLYFO(ICHT3+4,ELNUM)
AM3 = ELLYFO(ICHT3+5,ELNUM)
AL4 = ELLYFO(ICHT3+6,ELNUM)
AM4 = ELLYFO(ICHT3+7,ELNUM)
AL5 = ELLYFO(ICHT3+8,ELNUM)
ICOUNT(ISTOP3,ILAYER) = ICOUNT(ISTOP3,ILAYER) + 1
IPHT = ICOUNT(ISTOP3,ILAYER)
TEHSTF(ISTOP3,3*IPHT,ILAYER) = IPHT
TEHSTF(ISTOP3,12*IPHT,ILAYER) = EPSY
TEHSTF(ISTOP3,63*IPHT,ILAYER) = ARATIO
TEHSTF(ISTOP3,60*IPHT,ILAYER) = SIGYLD
TEHSTF(ISTOP3,66*IPHT,ILAYER) = ILY
INT = 9*(IPHT-1) + 1
TEHSTF(ISTOP3,30*INT,ILAYER) = AL1
TEHSTF(ISTOP3,31*INT,ILAYER) = AM1
TEHSTF(ISTOP3,32*INT,ILAYER) = AL2
TEHSTF(ISTOP3,33*INT,ILAYER) = AM2
TEHSTF(ISTOP3,34*INT,ILAYER) = AL3
TEHSTF(ISTOP3,35*INT,ILAYER) = AM3
TEHSTF(ISTOP3,36*INT,ILAYER) = AL4
TEHSTF(ISTOP3,37*INT,ILAYER) = AM4
TEHSTF(ISTOP3,38*INT,ILAYER) = AL5
ENDIF
ELSE
ICHT3 = ICHT1 + 9
EPSY = EPSY1
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARAT01
AL1 = ELLYFO(ICHT3,ELNUM)
AM1 = ELLYFO(ICHT3+1,ELNUM)
AR1 = ELLYFO(ICHT3+2,ELNUM)
AL2 = ELLYFO(ICHT3+3,ELNUM)
AM2 = ELLYFO(ICHT3+4,ELNUM)
AR2 = ELLYFO(ICHT3+5,ELNUM)
AL3 = ELLYFO(ICHT3+6,ELNUM)
AM3 = ELLYFO(ICHT3+7,ELNUM)
AR3 = ELLYFO(ICHT3+8,ELNUM)
ICOUNT(INTGPE,ILAYER) = ICOUNT(INTGPE,ILAYER) + 1
IPBT = ICOUNT(INTGPE,ILAYER)
TEHSTF(INTGPE,3*IPBT,ILAYER) = IPBT
TEHSTF(INTGPE,12*IPBT,ILAYER) = EPSY
TEHSTF(INTGPE,66*IPBT,ILAYER) = ILY
TEHSTF(INTGPE,60*IPBT,ILAYER) = ARATIO
TEHSTF(INTGPE,63*IPBT,ILAYER) = SIGYLD
IMT = 9*(IPBT-1) + 1
TEHSTF(INTGPE,30+IMT,ILAYER) = AL1
TEHSTF(INTGPE,31+IMT,ILAYER) = AM1
TEHSTF(INTGPE,32+IMT,ILAYER) = AR1
TEHSTF(INTGPE,33+IMT,ILAYER) = AL2
TEHSTF(INTGPE,34+IMT,ILAYER) = AM2
TEHSTF(INTGPE,35+IMT,ILAYER) = AR2
TEHSTF(INTGPE,36+IMT,ILAYER) = AL3
TEHSTF(INTGPE,37+IMT,ILAYER) = AM3
TEHSTF(INTGPE,38+IMT,ILAYER) = AR3
ENDIF
ELSE
ICHT3 = ICHT1 + 9
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARAT01
AL1 = ELLYFO(ICHT3,ELNUM)
AM1 = ELLYFO(ICHT3+1,ELNUM)
AR1 = ELLYFO(ICHT3+2,ELNUM)
AL2 = ELLYFO(ICHT3+3,ELNUM)
AM2 = ELLYFO(ICHT3+4,ELNUM)
AR2 = ELLYFO(ICHT3+5,ELNUM)
AL3 = ELLYFO(ICHT3+6,ELNUM)
AM3 = ELLYFO(ICHT3+7,ELNUM)
AR3 = ELLYFO(ICHT3+8,ELNUM)
ICOUNT(INTGPE,ILAYER) = ICOUNT(INTGPE,ILAYER) + 1
IPBT = ICOUNT(INTGPE,ILAYER)
TEHSTF(INTGPE,3*IPBT,ILAYER) = IPBT
TEHSTF(INTGPE,66*IPBT,ILAYER) = ILY
TEHSTF(INTGPE,60*IPBT,ILAYER) = EPSY
TEHSTF(INTGPE,63*IPBT,ILAYER) = SIGYLD
IMT = 9*(IPBT-1) + 1
TEHSTF(INTGPE,30+IMT,ILAYER) = AL1
TEHSTF(INTGPE,31+IMT,ILAYER) = AM1
TEHSTF(INTGPE,32+IMT,ILAYER) = AR1
TEHSTF(INTGPE,33+IMT,ILAYER) = AL2
TEHSTF(INTGPE,34+IMT,ILAYER) = AM2
TEHSTF(INTGPE,35+IMT,ILAYER) = AR2
TEHSTF(INTGPE,36+IMT,ILAYER) = AL3
TEHSTF(INTGPE,37+IMT,ILAYER) = AM3

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
TEHSTF(INTGPH,36+IMT,ILAYER) = A3
ENDIF
ELSE
ICHT3 = ICHT1 + 9
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARAT10
AL1 = ELLYFO(ICHT3,ELNUM)
AM1 = ELLYFO(ICHT3+1,ELNUM)
AL1 = ELLYFO(ICHT3+2,ELNUM)
AL2 = ELLYFO(ICHT3+3,ELNUM)
AM2 = ELLYFO(ICHT3+4,ELNUM)
AL3 = ELLYFO(ICHT3+5,ELNUM)
AM3 = ELLYFO(ICHT3+6,ELNUM)
AM3 = ELLYFO(ICHT3+7,ELNUM)
AM3 = ELLYFO(ICHT3+8,ELNUM)
ICOUHT(IHTGPH,ILAYER) = ICOUHT(IHTGPH,ILAYER)
IPH T = ICOUHT(IHTGPH,ILAYER)
TEHSTF(INTGPH,12+IPHT,ILAYER) = IPHT
TEHSTF(INTGPH,12+IPHT,ILAYER) = EPSY
TEHSTF(INTGPH,60+IPHT,ILAYER) = EPSY
TEHSTF(INTGPH,60+IPHT,ILAYER) = ARATIO
TEHSTF(INTGPH,60+IPHT,ILAYER) = SIGYLD
IMT = 9*(IPHT-1) + 1
TEHSTF(INTGPH,30+IMT,ILAYER) = AL1
TEHSTF(INTGPH,31+IMT,ILAYER) = AM1
TEHSTF(INTGPH,32+IMT,ILAYER) = AL1
TEHSTF(INTGPH,33+IMT,ILAYER) = AL2
TEHSTF(INTGPH,34+IMT,ILAYER) = AM2
TEHSTF(INTGPH,35+IMT,ILAYER) = AM2
TEHSTF(INTGPH,36+IMT,ILAYER) = AM3
TEHSTF(INTGPH,37+IMT,ILAYER) = AM3
TEHSTF(INTGPH,38+IMT,ILAYER) = AM3
ENDIF
ENDIF
ENDIF
ENDIF
IF (ICOUNT(INTGPH,ILAYER) .LE. 2) THEN
IF (ILAYER .LE. ILAYRS .AND. ILAYRS.GE.1) THEN
ICOH = (ILAYER-1)*21 + 1
MAC = ELLYFO(ICOH,ELNUM)
ES = PROPER(1,MAC)
ICHT1 = (ILAYER+1)*21 + 1
MATRL1 = ELLYFO(ICHT1,ELNUM)
ICK1 = PROPER(26,MATRL1)
IF (ICK1 .EQ. 0) THEN
ILY1 = ILAYER + 2
EPSY1 = PROPER(6,MATRL1)/PROPER(1,MATRL1)
SIGYD1 = PROPER(5,MATRL1)
ARAT1 = PROPER(1,MATRL1)/ES
ICHT2 = ILAYER*21 + 1
MATRL2 = ELLYFO(ICHT2,ELNUM)
ICK2 = PROPER(26,MATRL2)
IF (ICK2 .EQ. 0) THEN
IF (ILAYER .EQ. 0) THEN
ICHT3 = (ILAYER-2)*21 + 1
MATRL3 = ELLYFO(ICHT3,ELNUM)
ICK3 = PROPER(26,MATRL3)
ENDIF
ENDIF
ENDIF
ENDIF
IF (IC Ge. 0) THEN
  ILY3 = ILAYER - 1
  EPSY3 = PROPER(6, MATRL3)/PROPER(1, MATRL3)
  SIGY3 = PROPER(6, MATRL3)
  ARATIO3 = PROPER(1, MATRL3)/ES
  ICST4 = ICST1 + 9
  ICST6 = ICST3 + 9
  AL1 = ELLYFO(ICST4, ELNUM)
  AM1 = ELLYFO(ICST4+1, ELNUM)
  AB1 = ELLYFO(ICST4+2, ELNUM)
  AL2 = ELLYFO(ICST6, ELNUM)
  AM2 = ELLYFO(ICST6+1, ELNUM)
  AN2 = ELLYFO(ICST6+2, ELNUM)
  DOT = AL1*AL2 + AM1*AM2 + AB1*AN2
  IF (DOT .GE. 0.999 .AND. DOT .LE. 1.001) THEN
    EPSY = EPSY3
    ILY = ILY3
    SIGYLD = SIGY3
    ARATIO = ARATIO3
    AL1 = ELLYFO(ICST4, ELNUM)
    AM1 = ELLYFO(ICST4+1, ELNUM)
    AB1 = ELLYFO(ICST4+2, ELNUM)
    AL2 = ELLYFO(ICST6, ELNUM)
    AM2 = ELLYFO(ICST6+1, ELNUM)
    AN2 = ELLYFO(ICST6+2, ELNUM)
  ELSE
    EPSY = EPSY1
    ILY = ILY1
    SIGYLD = SIGY1
    ARATIO = ARATIO1
    AL1 = ELLYFO(ICST4, ELNUM)
    AM1 = ELLYFO(ICST4+1, ELNUM)
    AB1 = ELLYFO(ICST4+2, ELNUM)
    AL2 = ELLYFO(ICST6, ELNUM)
    AM2 = ELLYFO(ICST6+1, ELNUM)
    AN2 = ELLYFO(ICST6+2, ELNUM)
  ENDIF
  ICOUNT(IGTPH, ILAYER) = ICOUNT(IGTPH, ILAYER) + 1
  IPST = ICOUNT(IGTPH, ILAYER)
  TESSTF(IGTPH, 4, IPST, ILAYER) = IPST
  TESSTF(IGTPH, 12, IPST, ILAYER) = EPSY
  TESSTF(IGTPH, 66, IPST, ILAYER) = ILY
  TESSTF(IGTPH, 60, IPST, ILAYER) = ARATIO
  TESSTF(IGTPH, 63, IPST, ILAYER) = SIGYLD
  IMT = 9*(IPST-1) + 1
  TESSTF(IGTPH, 30+IMT, ILAYER) = AL1
  TESSTF(IGTPH, 31+IMT, ILAYER) = AM1
  TESSTF(IGTPH, 32+IMT, ILAYER) = AB1
  TESSTF(IGTPH, 33+IMT, ILAYER) = AL2
  TESSTF(IGTPH, 34+IMT, ILAYER) = AM2
  TESSTF(IGTPH, 35+IMT, ILAYER) = AN2
  TESSTF(IGTPH, 36+IMT, ILAYER) = AL3
  TESSTF(IGTPH, 37+IMT, ILAYER) = AM3
  TESSTF(IGTPH, 38+IMT, ILAYER) = AN3
ELSE IF (ILAYER - 2 .GT. 0) THEN
    ICST6 = (ILAYER-3)*21 + 1
    MATRL4 = ELLYFO(ICST6,ELUM)
    ICK4 = PROPER(26,MATRL4)
    IF (ICK4 .EQ. 0) THEN
        ILY4 = ILAYER - 2
        EPSY4 = PROPER(1,MATRL4)/PROPER(1,
        MATRL4)
        ARAT04 = PROPER(1,MATRL4)/ES
        SIGYD4 = PROPER(5,MATRL4)
        ICST3 = ICST1 + 9
        ICST6 = ICST + 9
        AL1 = ELLYFO(ICST3,ELUM)
        AM1 = ELLYFO(ICST3+1,ELUM)
        AS1 = ELLYFO(ICST3+2,ELUM)
        AL2 = ELLYFO(ICST6,ELUM)
        AM2 = ELLYFO(ICST6+1,ELUM)
        AS2 = ELLYFO(ICST6+2,ELUM)
        DOT = AL1*AL2 + AM1*AS2 + AS1*AS1
        IF (DABS(DOT).GE.0.999 .AND. DABS(DOT)
        .LE.1.001) THEN
            IF (EPSY4 .GT. EPSY1) THEN
                EPSY = EPSY4
                ILY = ILY4
                SIGYLD = SIGYD4
                ARATIO = ARAT04
            ELSE
                EPSY = EPSY1
                ILY = ILY1
                ARATIO = ARAT01
                SIGYLD = SIGYD1
            ENDIF
            ICOUNT(INTOPS,ILAYER) = ICOUNT(INTOPS,
            ILAYER) + 1
            IPST = ICOUNT(INTOPS,ILAYER)
            TESSTF(INTOPS,3+IPST,ILAYER) = IPST
            TESSTF(INTOPS,12+IPST,ILAYER) = EPSY
            TESSTF(INTOPS,60+IPST,ILAYER) = ARAT01
            TESSTF(INTOPS,66+IPST,ILAYER) = ILY
            TESSTF(INTOPS,63+IPST,ILAYER) = SIGYD1
            INT = 9*(IPST-1) + 1
            TESSTF(INTOPS,30+INT,ILAYER) = AL1
            TESSTF(INTOPS,31+INT,ILAYER) = AM1
            TESSTF(INTOPS,32+INT,ILAYER) = AS1
            TESSTF(INTOPS,33+INT,ILAYER) = AL2
        ELSE
            EPSY = EPSY1
            ILY = ILY1
            ARATIO = ARAT01
            SIGYLD = SIGYD1
            ICOUNT(INTOPS,ILAYER) = ICOUNT(INTOPS,
            ILAYER)
        ENDIF
    ELSE
        EPSY = EPSY1
        ILY = ILY1
        ARATIO = ARAT01
        SIGYLD = SIGYD1
        ICOUNT(INTOPS,ILAYER) = ICOUNT(INTOPS,
        ILAYER)
    ENDIF
ELSE
ICES = ICET1 + 9
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARAT01
AL1 = ELLYFO(ICES3,ELNUM)
AM1 = ELLYFO(ICES3+1,ELNUM)
AL2 = ELLYFO(ICES3+2,ELNUM)
AM2 = ELLYFO(ICES3+3,ELNUM)
AL3 = ELLYFO(ICES3+4,ELNUM)
AM3 = ELLYFO(ICES3+5,ELNUM)
AL4 = ELLYFO(ICES3+6,ELNUM)
AM4 = ELLYFO(ICES3+7,ELNUM)
AL5 = ELLYFO(ICES3+8,ELNUM)
AM5 = ELLYFO(ICES3+9,ELNUM)
ICOUT = ICOUT(IPTS,ILAYER) + 1
IPST = ICOUT(IPOS,ILAYER)
INST = INST + 1
ELSE
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARAT01
ICES = ICET1 + 9
AL1 = ELLYFO(ICES3,ELNUM)
AM1 = ELLYFO(ICES3+1,ELNUM)
AL2 = ELLYFO(ICES3+2,ELNUM)
AM2 = ELLYFO(ICES3+3,ELNUM)
AL3 = ELLYFO(ICES3+4,ELNUM)
AM3 = ELLYFO(ICES3+5,ELNUM)
AL4 = ELLYFO(ICES3+6,ELNUM)
AM4 = ELLYFO(ICES3+7,ELNUM)
AL5 = ELLYFO(ICES3+8,ELNUM)
AM5 = ELLYFO(ICES3+9,ELNUM)
ICOUT = ICOUT(IPOS,ILAYER) + 1
IPST = ICOUT(IPOS,ILAYER)
INST = INST + 1
ENDIF
TEHSTF (IH TG PH, 30+IM T, ILAYER) = AL1
TEHSTF (IH TG PH, 31+IM T, ILAYER) = AM1
TEHSTF (IH TG PH, 32+IM T, ILAYER) = AH1
TEHSTF (IH TG PH, 33+IM T, ILAYER) = AL2
TEHSTF (IH TG PH, 34+IM T, ILAYER) = AM2
TEHSTF (IH TG PH, 35+IM T, ILAYER) = AH2
TEHSTF (IH TG PH, 36+IM T, ILAYER) = AL3
TEHSTF (IH TG PH, 37+IM T, ILAYER) = AM3
TEHSTF (IH TG PH, 38+IM T, ILAYER) = AH3

ENDIF
ELSE
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARAT01
ICBT3 = ICBT1 + 9
AL1 = ELLYFO (ICBT3, ELNUM)
AM1 = ELLYFO (ICBT3+1, ELNUM)
AH1 = ELLYFO (ICBT3+2, ELNUM)
AL2 = ELLYFO (ICBT3+3, ELNUM)
AM2 = ELLYFO (ICBT3+4, ELNUM)
AH2 = ELLYFO (ICBT3+5, ELNUM)
AL3 = ELLYFO (ICBT3+6, ELNUM)
AM3 = ELLYFO (ICBT3+7, ELNUM)
AH3 = ELLYFO (ICBT3+8, ELNUM)
ICOUNT (IH TG PH, ILAYER) = ICOUHT (IH TG PH, ILAYER) + 1

IPHT = ICOUHT (IH TG PH, ILAYER)
TEHSTF (IH TG PH, 30+IPHT, ILAYER) = EPST
TEHSTF (IH TG PH, 31+IPHT, ILAYER) = EPSY
TEHSTF (IH TG PH, 32+IPHT, ILAYER) = ILY
TEHSTF (IH TG PH, 33+IPHT, ILAYER) = ARATIO
TEHSTF (IH TG PH, 34+IPHT, ILAYER) = SIGYLD

IMT = 9*(IPHT-1) + 1
TEHSTF (IH TG PH, 30+IMT, ILAYER) = AL1
TEHSTF (IH TG PH, 31+IMT, ILAYER) = AM1
TEHSTF (IH TG PH, 32+IMT, ILAYER) = AH1
TEHSTF (IH TG PH, 33+IMT, ILAYER) = AL2
TEHSTF (IH TG PH, 34+IMT, ILAYER) = AM2
TEHSTF (IH TG PH, 35+IMT, ILAYER) = AH2
TEHSTF (IH TG PH, 36+IMT, ILAYER) = AL3
TEHSTF (IH TG PH, 37+IMT, ILAYER) = AM3
TEHSTF (IH TG PH, 38+IMT, ILAYER) = AH3

ENDIF
ELSE
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARAT01
ICBT3 = ICBT1 + 9
AL1 = ELLYFO (ICBT3, ELNUM)
AM1 = ELLYFO (ICBT3+1, ELNUM)
AH1 = ELLYFO (ICBT3+2, ELNUM)
AL2 = ELLYFO (ICBT3+3, ELNUM)
AM2 = ELLYFO (ICBT3+4, ELNUM)
AH2 = ELLYFO (ICBT3+5, ELNUM)
AL3 = ELLYFO (ICBT3+6, ELNUM)
AM3 = ELLYFO (ICBT3+7, ELNUM)
AH3 = ELLYFO (ICBT3+8, ELNUM)
ICOUNT (IH TG PH, ILAYER) = ICOUHT (IH TG PH, ILAYER) + 1

IPHT = ICOUHT (IH TG PH, ILAYER)
ENDIF

ELSE

EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYLD1
ARATIO = ARATIO1
ICH T3 = ICH T1 + 9
AL1 = ELLYFO(ICT3,ELEM)
AM1 = ELLYFO(ICT3+1,ELEM)
AL2 = ELLYFO(ICT3+2,ELEM)
AM2 = ELLYFO(ICT3+3,ELEM)
AL3 = ELLYFO(ICT3+4,ELEM)
AM3 = ELLYFO(ICT3+5,ELEM)
AH3 = ELLYFO(ICT3+6,ELEM)
ICOUNT(ISTOP,ILAYER) = ICOUNT(ISTOP,ILAYER) + 1

IPST = ICOUNT(ISTOP,ILAYER)
TESTSF(ISTOP,3+IPST,ILAYER) = IPST
TESTSF(ISTOP,12+IPST,ILAYER) = EPSY
TESTSF(ISTOP,66+IPST,ILAYER) = ILY
TESTSF(ISTOP,63+IPST,ILAYER) = ARATIO
TESTSF(ISTOP,60+IPST,ILAYER) = SIGYLD
INT = 9*(IPST-1) + 1
TESTSF(ISTOP,90+INT,ILAYER) = AL1
TESTSF(ISTOP,91+INT,ILAYER) = AM1
TESTSF(ISTOP,92+INT,ILAYER) = AL2
TESTSF(ISTOP,93+INT,ILAYER) = AM2
TESTSF(ISTOP,94+INT,ILAYER) = AL3
TESTSF(ISTOP,95+INT,ILAYER) = AM3
TESTSF(ISTOP,96+INT,ILAYER) = AS3
ENDIF

ENDIF

ENDIF

ENDIF

C IF (ICOUNT(ISTOP,ILAYER) .LE. 2) THEN
IF (ILAYER - 1.6E-1 .ABD. ILAYER.LE.ILAYS) THEN
ICON = (ILAYER-1)*21 + 1
MAC = ELLYFO(ICON,ELEM)
ES = PROPER(1,MAC)
ICT1 = (ILAYER-2)*21 + 1
MATRL1 = ELLYFO(ICT1,ELEM)
ICK1 = PROPER(28,MATRL1)
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
IF (ICK1 .EQ. 0) THEN
EPSY1 = PROPER(6,MATRL1)/PROPER(1,MATRL1)
SIGYD1 = PROPER(6,MATRL1)
ARATOl = PROPER(1,MATRL1)/ES
ILYL = ILAGER - 1
IF (ILAYER + 1 .LE. LAYERS) THEN
ICHT2 = ILAGER*21 + 1
MATRL2 = ELLYFO(ICHT2,ELHUB)
ICK2 = PROPER(26,MATRL2)
IF (ICK2 .EQ. 0) THEN
EPSY2 = PROPER(6,MATRL2)/PROPER(1,MATRL2)
SIGYD2 = PROPER(6,MATRL2)
ARAT02 = PROPER(1,MATRL2)/ES
ILYL2 = ILAGER + 1
ICHT3 = ICHT1 + 9
ICHT4 = ICHT2 + 9
AL1 = ELLYFO(ICHT3,ELHUB)
AM1 = ELLYFO(ICHT3+1,ELHUB)
AB1 = ELLYFO(ICHT3+2,ELHUB)
AL2 = ELLYFO(ICHT4,ELHUB)
AM2 = ELLYFO(ICHT4+1,ELHUB)
AB2 = ELLYFO(ICHT4+2,ELHUB)
AL3 = ELLYFO(ICHT4+3,ELHUB)
AM3 = ELLYFO(ICHT4+4,ELHUB)
AB3 = ELLYFO(ICHT4+5,ELHUB)
DOT = AL1*AL2 + AB1*AB2 + AH1*AH2
IF (DABS(DOT).GE.0.999 .AND. DABS(DOT).LE.1.001) THEN
IF (EPSY2 .GT. EPSY1) THEN
EPSY = EPSY2
ILYL = ILY2
SIGYDLD = SIGYD2
ARATIO = ARAT02
ELSE
EPSY = EPSY1
ILYL = ILY1
SIGYDLD = SIGYD1
ARATIO = ARAT01
ENDIF
ICHT = ICHT + 9
ENDIF

1

1

1
ICOUNT(ISTOPS, ILAYER) = ICOUNT(ISTOPS, ILAYER) + 1
IPRT = ICOUNT(ISTOPS, ILAYER)
TEHSTF(IIGTGP,3*IPHT,ILAYER) = IPHT
TEHSTF(IIGTGP,12*IPHT,ILAYER) = EPSY
TEHSTF(IIGTGP,63*IPHT,ILAYER) = SIGYLD
TEHSTF(IIGTGP,66*IPHT,ILAYER) = ILY
TEHSTF(IIGTGP,60*IPHT,ILAYER) = ARATIO
IMT = 9*(IPHT-1) + 1
TEHSTF(IIGTGP,30+IMT,ILAYER) = AL1
TEHSTF(IIGTGP,31+IMT,ILAYER) = AM1
TEHSTF(IIGTGP,32+IMT,ILAYER) = AR1
TEHSTF(IIGTGP,33+IMT,ILAYER) = AL2
TEHSTF(IIGTGP,34+IMT,ILAYER) = AM2
TEHSTF(IIGTGP,36+IMT,ILAYER) = AR2
TEHSTF(IIGTGP,36+IMT,ILAYER) = AL3
TEHSTF(IIGTGP,37+IMT,ILAYER) = AM3
TEHSTF(IIGTGP,38+IMT,ILAYER) = AR3
ELSE
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARATIO1
AL1 = ELLYFO(IICT3+1,ELSUM)
AM1 = ELLYFO(IICT3+1,ELSUM)
AR1 = ELLYFO(IICT3+2,ELSUM)
AL2 = ELLYFO(IICT3+3,ELSUM)
AM2 = ELLYFO(IICT3+4,ELSUM)
AR2 = ELLYFO(IICT3+5,ELSUM)
AL3 = ELLYFO(IICT3+6,ELSUM)
AM3 = ELLYFO(IICT3+7,ELSUM)
AR3 = ELLYFO(IICT3+8,ELSUM)
ICOUNT(IIGTGP,ILAYER) = ICOUNT(IIGTGP,ILAYER) + 1
IPHT = ICOUNT(IIGTGP,ILAYER)
TEHSTF(IIGTGP,3*IPHT,ILAYER) = IPHT
TEHSTF(IIGTGP,12*IPHT,ILAYER) = EPSY
TEHSTF(IIGTGP,63*IPHT,ILAYER) = SIGYLD
TEHSTF(IIGTGP,66*IPHT,ILAYER) = ILY
TEHSTF(IIGTGP,60*IPHT,ILAYER) = ARATIO
IMT = 9*(IPHT-1) + 1
TEHSTF(IIGTGP,30+IMT,ILAYER) = AL1
TEHSTF(IIGTGP,31+IMT,ILAYER) = AM1
TEHSTF(IIGTGP,32+IMT,ILAYER) = AR1
TEHSTF(IIGTGP,33+IMT,ILAYER) = AL2
TEHSTF(IIGTGP,34+IMT,ILAYER) = AM2
TEHSTF(IIGTGP,36+IMT,ILAYER) = AR2
TEHSTF(IIGTGP,36+IMT,ILAYER) = AL3
TEHSTF(IIGTGP,37+IMT,ILAYER) = AM3
TEHSTF(IIGTGP,38+IMT,ILAYER) = AR3
ENDIF
ELSE
ICT3 = ICT3 + 1
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARATIO1
AL1 = ELLYFO(IICT3,ELSUM)
AM1 = ELLYFO(IICT3+1,ELSUM)
AR1 = ELLYFO(IICT3+2,ELSUM)
AL2 = ELLYFO(IICT3+3,ELSUM)
AM2 = ELLYFO(IICT3+4,ELSUM)
AR2 = ELLYFO(IICT3+5,ELSUM)
AL3 = ELLYFO(IICT3+6,ELSUM)
AM3 = ELLYFO(IICT3+7,ELSUM)
AR3 = ELLYFO(IICT3+8,ELSUM)
ICOUNT(ICTGPH,ILAYER) = ICBUCT(ICTGPH, ILAYER) + 1
IPST = ICBUCT(ICTGPH,ILAYER)
TENSIF(ICTGPH,3*IPST,ILAYER) = IPST
TENSIF(ICTGPH,12*IPST,ILAYER) = EPSY
TENSIF(ICTGPH,66*IPST,ILAYER) = ARATIO
TENSIF(ICTGPH,63*IPST,ILAYER) = SIGYLD
TENSIF(ICTGPH,60*IPST,ILAYER) = ILY
INT = 9*(IPST-1) + 1
TENSIF(ICTGPH,30*INT,ILAYER) = AL1
TENSIF(ICTGPH,31*INT,ILAYER) = AM1
TENSIF(ICTGPH,32*INT,ILAYER) = AL2
TENSIF(ICTGPH,33*INT,ILAYER) = AM2
TENSIF(ICTGPH,34*INT,ILAYER) = AL3
TENSIF(ICTGPH,36*INT,ILAYER) = AM3
TENSIF(ICTGPH,37*INT,ILAYER) = AM3
TENSIF(ICTGPH,39*INT,ILAYER) = AM3

ENDIF
ELSE
ICHT3 = ICT1 + 9
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARATD1
AL1 = ELLYFO(ICHT3,ELNUM)
AM1 = ELLYFO(ICHT3+1,ELNUM)
AL2 = ELLYFO(ICHT3+2,ELNUM)
AM2 = ELLYFO(ICHT3+3,ELNUM)
AL3 = ELLYFO(ICHT3+4,ELNUM)
AM3 = ELLYFO(ICHT3+5,ELNUM)
AM3 = ELLYFO(ICHT3+6,ELNUM)

ICOUNT(ICTGPH,ILAYER) = ICBUCT(ICTGPH, ILAYER) + 1
IPST = ICBUCT(ICTGPH,ILAYER)
TENSIF(ICTGPH,3*IPST,ILAYER) = IPST
TENSIF(ICTGPH,12*IPST,ILAYER) = EPSY
TENSIF(ICTGPH,66*IPST,ILAYER) = ILY
TENSIF(ICTGPH,63*IPST,ILAYER) = SIGYLD
INT = 9*(IPST-1) + 1
TENSIF(ICTGPH,30*INT,ILAYER) = AL1
TENSIF(ICTGPH,31*INT,ILAYER) = AM1
TENSIF(ICTGPH,32*INT,ILAYER) = AL2
TENSIF(ICTGPH,33*INT,ILAYER) = AM2
TENSIF(ICTGPH,34*INT,ILAYER) = AL3
TENSIF(ICTGPH,36*INT,ILAYER) = AM3
TENSIF(ICTGPH,37*INT,ILAYER) = AM3

ENDIF
ELSE
ICHT3 = ICT1 + 9
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARATD1
AL1 = ELLYFO(ICHT3,ELNUM)
AM1 = ELLYFO(ICHT3+1,ELNUM)
AM1 = ELLYFO(ICHT3+2,ELNUM)
AL2 = ELLYFO(ICHT3+3,ELNUM)
AL3 = ELLYFO(ICHT3+4,ELNUM)
AM3 = ELLYFO(ICHT3+5,ELNUM)
AM3 = ELLYFO(ICHT3+6,ELNUM)

ENDIF

!B 2  = ELLYFO (ICTH3 + 4, ELHUM)
AL3 = ELLYFO (ICTH3 + 6, ELHUM)
AM3 = ELLYFO (ICTH3 + 7, ELHUM)
A43 = ELLYFO (ICTH3 + 8, ELHUM)

ICOUNT (INTGPH, LAYER) = ICOUNT (INTGPH, LAYER) + 1

IPHT = ICOUNT (INTGPH, LAYER)
TENSTF (INTGPH, 3 + IPHT, LAYER) = IPHT
TENSTF (INTGPH, 5 + IPHT, LAYER) = EPSY
TENSTF (INTGPH, 12 + IPHT, LAYER) = ILY
TENSTF (INTGPH, 60 + IPHT, LAYER) = ARATIO
TENSTF (INTGPH, 63 + IPHT, LAYER) = SIGYLD

IMT = 9 * (IPHT - 1) + 1
TENSTF (INTGPH, 30 + IMT, LAYER) = AL1
TENSTF (INTGPH, 31 + IMT, LAYER) = AM1
TENSTF (INTGPH, 32 + IMT, LAYER) = AL2
TENSTF (INTGPH, 33 + IMT, LAYER) = AM2
TENSTF (INTGPH, 34 + IMT, LAYER) = AL3
TENSTF (INTGPH, 35 + IMT, LAYER) = AM3
TENSTF (INTGPH, 36 + IMT, LAYER) = AL4
TENSTF (INTGPH, 37 + IMT, LAYER) = AM4
TENSTF (INTGPH, 38 + IMT, LAYER) = AL5

ENDIF

ELSE
ICTH3 = ICTH1 + 9
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYLD1
ARATIO = ARAT01
AL1 = ELLYFO (ICTH3, ELHUM)
AM1 = ELLYFO (ICTH3 + 1, ELHUM)
AL2 = ELLYFO (ICTH3 + 2, ELHUM)
AM2 = ELLYFO (ICTH3 + 3, ELHUM)
AL3 = ELLYFO (ICTH3 + 4, ELHUM)
AM3 = ELLYFO (ICTH3 + 5, ELHUM)

ICOUNT (INTGPH, LAYER) = ICOUNT (INTGPH, LAYER) + 1

IPHT = ICOUNT (INTGPH, LAYER)
TENSTF (INTGPH, 3 + IPHT, LAYER) = IPHT
TENSTF (INTGPH, 12 + IPHT, LAYER) = EPSY
TENSTF (INTGPH, 60 + IPHT, LAYER) = ILY
TENSTF (INTGPH, 63 + IPHT, LAYER) = ARATIO
TENSTF (INTGPH, 63 + IPHT, LAYER) = SIGYLD

IMT = 9 * (IPHT - 1) + 1
TENSTF (INTGPH, 30 + IMT, LAYER) = AL1
TENSTF (INTGPH, 31 + IMT, LAYER) = AM1
TENSTF (INTGPH, 32 + IMT, LAYER) = AL2
TENSTF (INTGPH, 33 + IMT, LAYER) = AM2
TENSTF (INTGPH, 34 + IMT, LAYER) = AL3
TENSTF (INTGPH, 35 + IMT, LAYER) = AM3
TENSTF (INTGPH, 36 + IMT, LAYER) = AL4
TENSTF (INTGPH, 37 + IMT, LAYER) = AM4
TENSTF (INTGPH, 38 + IMT, LAYER) = AL5

ENDIF

ENDIF
IF (ICONST(ISTGFS,ILAYER) .LE. 2) THEN
  IF (ILAYER - 2.0E-1 .ASD. ILAYER.LE.ILAYRS) THEN
    ICGB = (ILAYER-1)*21 + 1
    MAC = ELLYFO(ICGB,ELNUM)
    ES = PROPER(1,MAC)
    ICBT1 = (ILAYER-3)*21 + 1
    MATRL1 = ELLYFO(ICBT1,ELNUM)

    ICK1 = PROPER(25,MATRL1)
    IF (ICK1 .EQ. 0) THEN
      ILY1 = ILAYER - 2
      EPSY1 = PROPER(5,MATRL1)/PROPER(1,MATRL1)
      SIGYD1 = PROPER(5,MATRL1)
      ARATIO1 = PROPER(1,MATRL1)/ES
      ICBT2 = (ILAYER-2)*21 + 1
      MATRL2 = ELLYFO(ICBT2,ELNUM)
      ICK2 = PROPER(25,MATRL2)
      IF (ICK2 .EQ. 0) THEN
        IF (ILAYER + 1 .LE. ILAYRS) THEN
          ICBT3 = ILAYER*21 + 1
          MATRL3 = ELLYFO(ICBT3,ELNUM)
          ICK3 = PROPER(25,MATRL3)
          IF (ICK3 .EQ. 0) THEN
            ILY3 = ILAYER + 1
            EPSY3 = PROPER(5,MATRL3)/PROPER(1,MATRL3)
            SIGYD3 = PROPER(5,MATRL3)
            ARATIO3 = PROPER(1,MATRL3)/ES
            ICBT4 = ICHT1 + 9
            ICBT5 = ICHT3 + 9
            AL1 = ELLYFO(ICBT4,ELNUM)
            AM1 = ELLYFO(ICBT4+1,ELNUM)
            AN1 = ELLYFO(ICBT4+2,ELNUM)
            AL2 = ELLYFO(ICBT5,ELNUM)
            AM2 = ELLYFO(ICBT5+1,ELNUM)
            AN2 = ELLYFO(ICBT5+2,ELNUM)
            DOT = AL1+AL2 + AM1+AM2 + AN1+AN2
            IF (DABS(DOT) .GE. 0.999 .AND. DABS(DOT) .LE. 1.000) THEN
              IF (EPSY3 .GT. EPSY1) THEN
                EPSY = EPSY3
              ELSE
                EPSY = EPSY1
              END IF
          ELSE
            EPSY = EPSY1
            ILY = ILY1
          END IF
        ELSE
          EPSY = EPSY1
        END IF
      END IF
    END IF
  END IF
END IF
$\text{AL}_2 = \text{ELLYFO}(\text{ICHT}4+6,\text{ELNUM})$

$\text{AM}_3 = \text{ELLYFO}(\text{ICHT}4+7,\text{ELNUM})$

$\text{AR}_3 = \text{ELLYFO}(\text{ICHT}4+8,\text{ELNUM})$

ENDIF

$\text{ICOUNT} (\text{INTOPN}, \text{ILAYER}) = \text{ICOUNT} (\text{INTOPN}, \text{ILAYER}) + 1$

$\text{IPHT} = \text{ICOUNT} (\text{INTOPN}, \text{ILAYER})$

$\text{TEHSTF} (\text{INTOPN}, 3+\text{IPHT}, \text{ILAYER}) = \text{IPHT}$

$\text{TEHSTF} (\text{INTOPN}, 12+\text{IPHT}, \text{ILAYER}) = \text{EPSY}$

$\text{TEHSTF} (\text{INTOPN}, 66+\text{IPHT}, \text{ILAYER}) = \text{ARATIO}$

$\text{TEHSTF} (\text{INTOPN}, 63+\text{IPHT}, \text{ILAYER}) = \text{SIGYLD}$

$\text{INT} = 9 \times (\text{IPHT}-1) + 1$

$\text{TEHSTF} (\text{INTOPN}, 30+\text{INT}, \text{ILAYER}) = \text{AL}_1$

$\text{TEHSTF} (\text{INTOPN}, 31+\text{INT}, \text{ILAYER}) = \text{AM}_1$

$\text{TEHSTF} (\text{INTOPN}, 32+\text{INT}, \text{ILAYER}) = \text{AH}_1$

$\text{TEHSTF} (\text{INTOPN}, 33+\text{INT}, \text{ILAYER}) = \text{AL}_2$

$\text{TEHSTF} (\text{INTOPN}, 34+\text{INT}, \text{ILAYER}) = \text{AM}_2$

$\text{TEHSTF} (\text{INTOPN}, 35+\text{INT}, \text{ILAYER}) = \text{AH}_2$

$\text{TEHSTF} (\text{INTOPN}, 36+\text{INT}, \text{ILAYER}) = \text{AL}_3$

$\text{TEHSTF} (\text{INTOPN}, 37+\text{INT}, \text{ILAYER}) = \text{AM}_3$

$\text{TEHSTF} (\text{INTOPN}, 38+\text{INT}, \text{ILAYER}) = \text{AH}_3$

ELSE IF (\text{ILAYER} + 2 .LE. \text{ILAYRS}) THEN

$\text{ICHT}_6 = (\text{ILAYER}+1) \times 21 + 1$

$\text{MATRL}_4 = \text{ELLYFO}(\text{ICHT}_6, \text{ELNUM})$

$\text{ICK}_4 = \text{PROPER}(26, \text{MATRL}_4)$

IF (\text{ICK}_4 .EQ. 0) THEN

$\text{ILY}_4 = \text{ILAYER} + 2$

$\text{EPSY}_4 = \text{PROPER}(8, \text{MATRL}_4)/\text{PROPER}(1, \text{MATRL}_4)$

$\text{ARAT0}_4 = \text{PROPER}(1, \text{MATRL}_4)/\text{ES}$

$\text{SIGYD}_4 = \text{PROPER}(6, \text{MATRL}_4)$

$\text{ICHT}_3 = \text{ICHT}_1 + 9$

$\text{ICHT}_6 = \text{ICHT}_6 + 9$

$\text{AL}_1 = \text{ELLYFO}(\text{ICHT}_3, \text{ELNUM})$

$\text{AM}_1 = \text{ELLYFO}(\text{ICHT}_3+1, \text{ELNUM})$

$\text{AH}_1 = \text{ELLYFO}(\text{ICHT}_3+2, \text{ELNUM})$

$\text{AL}_2 = \text{ELLYFO}(\text{ICHT}_6, \text{ELNUM})$

$\text{AM}_2 = \text{ELLYFO}(\text{ICHT}_6+1, \text{ELNUM})$

$\text{AH}_2 = \text{ELLYFO}(\text{ICHT}_6+2, \text{ELNUM})$

$\text{DOT} = \text{AL}_1*\text{AL}_2 + \text{AM}_1*\text{AM}_2 + \text{AH}_1*\text{AH}_2$

IF (DABS(DOT).GE.0.999 .AND. DABS(DOT).LE.1.001) THEN

IF (\text{EPSY}_4 .GT. \text{EPSY}_1) THEN

\text{EPSY} = \text{EPSY}_4

\text{ILY} = \text{ILY}_4

\text{SIGYLD} = \text{SIGYD}_4

\text{ARATIO} = \text{ARAT0}_4

\text{AL}_1 = \text{ELLYFO}(\text{ICHT}_6, \text{ELNUM})$

$\text{AM}_1 = \text{ELLYFO}(\text{ICHT}_6+1, \text{ELNUM})$

$\text{AH}_1 = \text{ELLYFO}(\text{ICHT}_6+2, \text{ELNUM})$

$\text{AL}_2 = \text{ELLYFO}(\text{ICHT}_6+3, \text{ELNUM})$

$\text{AM}_2 = \text{ELLYFO}(\text{ICHT}_6+4, \text{ELNUM})$

$\text{AH}_2 = \text{ELLYFO}(\text{ICHT}_6+5, \text{ELNUM})$

$\text{AL}_3 = \text{ELLYFO}(\text{ICHT}_6+6, \text{ELNUM})$

$\text{AM}_3 = \text{ELLYFO}(\text{ICHT}_6+7, \text{ELNUM})$

$\text{AR}_3 = \text{ELLYFO}(\text{ICHT}_6+8, \text{ELNUM})$

ELSE

\text{EPSY} = \text{EPSY}_1

\text{ILY} = \text{ILY}_1

\text{ARATIO} = \text{ARAT0}_1

\text{SIGYLD} = \text{SIGYD}_1

\text{AL}_1 = \text{ELLYFO}(\text{ICHT}_4, \text{ELNUM})$
AL1 = ELLYFO(ICT4+1,ELNUM)
AL2 = ELLYFO(ICT4+2,ELNUM)
AM2 = ELLYFO(ICT4+3,ELNUM)
AM1 = ELLYFO(ICT4+4,ELNUM)
AM2 = ELLYFO(ICT4+5,ELNUM)
AM3 = ELLYFO(ICT4+6,ELNUM)
AM3 = ELLYFO(ICT4+7,ELNUM)
AM3 = ELLYFO(ICT4+8,ELNUM)
ENDIF
ICOUNT(INTOPS,ILAYER) = ICOUNT(INTOPS, ILAYER) + 1
IPET = ICOUNT(INTOPS,ILAYER)
TGSTF(INTOPS,3+IPET,ILAYER) = IPET
TGSTF(INTOPS,12+IPET,ILAYER) = EPSY
TGSTF(INTOPS,60+IPET,ILAYER) = ARATIO
TGSTF(INTOPS,66+IPET,ILAYER) = SIGYLD
IMT = 9*(IPET-1) + 1
TGSTF(INTOPS,30+IMT,ILAYER) = AL1
TGSTF(INTOPS,31+IMT,ILAYER) = AM1
TGSTF(INTOPS,32+IMT,ILAYER) = AL2
TGSTF(INTOPS,33+IMT,ILAYER) = AM2
TGSTF(INTOPS,35+IMT,ILAYER) = AL3
TGSTF(INTOPS,36+IMT,ILAYER) = AM3
TGSTF(INTOPS,38+IMT,ILAYER) = AM3
ELSE
ICT3 = ICT3 + 9
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARAT01
AM1 = ELLYFO(ICT3+1,ELNUM)
AM1 = ELLYFO(ICT3+2,ELNUM)
AM2 = ELLYFO(ICT3+3,ELNUM)
AM2 = ELLYFO(ICT3+4,ELNUM)
AM3 = ELLYFO(ICT3+5,ELNUM)
AM3 = ELLYFO(ICT3+6,ELNUM)
AM3 = ELLYFO(ICT3+7,ELNUM)
AM3 = ELLYFO(ICT3+8,ELNUM)
ICOUNT(INTOPS,ILAYER) = ICOUNT(INTOPS, ILAYER) + 1
IPET = ICOUNT(INTOPS,ILAYER)
TGSTF(INTOPS,3+IPET,ILAYER) = IPET
TGSTF(INTOPS,12+IPET,ILAYER) = EPSY
TGSTF(INTOPS,60+IPET,ILAYER) = ARATIO
TGSTF(INTOPS,66+IPET,ILAYER) = SIGYLD
IMT = 9*(IPET-1) + 1
TGSTF(INTOPS,30+IMT,ILAYER) = AL1
TGSTF(INTOPS,31+IMT,ILAYER) = AM1
TGSTF(INTOPS,32+IMT,ILAYER) = AL2
TGSTF(INTOPS,33+IMT,ILAYER) = AM2
TGSTF(INTOPS,34+IMT,ILAYER) = AL3
TGSTF(INTOPS,36+IMT,ILAYER) = AM3
TGSTF(INTOPS,37+IMT,ILAYER) = AM3
ENDIF
ELSE
EPSY = EPSY1

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARAT01
ICHT3 = ICHT1 + 9
AL1 = ELLYFO(ICHT3,ELNUM)
AM1 = ELLYFO(ICHT3+1,ELNUM)
AL2 = ELLYFO(ICHT3+2,ELNUM)
AM2 = ELLYFO(ICHT3+3,ELNUM)
AH2 = ELLYFO(ICHT3+6,ELNUM)
AL3 = ELLYFO(ICHT3+6,ELNUM)
AM3 = ELLYFO(ICHT3+7,ELNUM)
AH3 = ELLYFO(ICHT3+9,ELNUM)
ICOUNT (INTOPN,ILAYER) = ICOUNT (INTOPN,ILAYER) + 1

IPST = ICOUNT (INTOPN,ILAYER)
TEHSTF (INTOPN,3*IPST,ILAYER) = EPSY
TEHSTF (INTOPN,12*IPST,ILAYER) = ARATIO
TEHSTF (INTOPN,60*IPST,ILAYER) = SIGYLD
IMT = 9*(IPST-1) + 1
TEHSTF (INTOPN,30+IMT,ILAYER) = AL1
TEHSTF (INTOPN,31+IMT,ILAYER) = AH1
TEHSTF (INTOPN,32+IMT,ILAYER) = AH1
TEHSTF (INTOPN,33+IMT,ILAYER) = AL2
TEHSTF (INTOPN,34+IMT,ILAYER) = AM2
TEHSTF (INTOPN,35+IMT,ILAYER) = AH2
TEHSTF (INTOPN,36+IMT,ILAYER) = AL3
TEHSTF (INTOPN,37+IMT,ILAYER) = AM3
TEHSTF (INTOPN,38+IMT,ILAYER) = AH3
ENDIF
ELSE
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARAT01
ICHT3 = ICHT1 + 9
AL1 = ELLYFO(ICHT3,ELNUM)
AM1 = ELLYFO(ICHT3+1,ELNUM)
AL2 = ELLYFO(ICHT3+2,ELNUM)
AM2 = ELLYFO(ICHT3+3,ELNUM)
AH2 = ELLYFO(ICHT3+6,ELNUM)
AL3 = ELLYFO(ICHT3+6,ELNUM)
AM3 = ELLYFO(ICHT3+7,ELNUM)
AH3 = ELLYFO(ICHT3+9,ELNUM)
ICOUNT (INTOPN,ILAYER) = ICOUNT (INTOPN,ILAYER) + 1

IPST = ICOUNT (INTOPN,ILAYER)
TEHSTF (INTOPN,3*IPST,ILAYER) = EPSY
TEHSTF (INTOPN,12*IPST,ILAYER) = ARATIO
TEHSTF (INTOPN,60*IPST,ILAYER) = SIGYLD
IMT = 9*(IPST-1) + 1
TEHSTF (INTOPN,30+IMT,ILAYER) = AL1
TEHSTF (INTOPN,31+IMT,ILAYER) = AM1
TEHSTF (INTOPN,32+IMT,ILAYER) = AH1
TEHSTF (INTOPN,33+IMT,ILAYER) = AL2
TEHSTF (INTOPN,34+IMT,ILAYER) = AM2
TEHSTF (INTOPN,35+IMT,ILAYER) = AH2
TENSTF(IHTGPH, 36+IMT, ILAYER) = AL3
TENSTF(IHTGPH, 37+IMT, ILAYER) = AM3
TENSTF(IHTGPH, 38+IMT, ILAYER) = AH3
ENDIF
ELSE
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARATOl
ICHT3 = ICHT1 + 9
AL1 = ELLYFO(ICHT3, ELHUM)
AM1 = ELLYFO(ICHT3+1, ELHUM)
AR1 = ELLYFO(ICHT3+2, ELHUM)
AL2 = ELLYFO(ICHT3+3, ELHUM)
AM2 = ELLYFO(ICHT3+4, ELHUM)
AR2 = ELLYFO(ICHT3+5, ELHUM)
AL3 = ELLYFO(ICHT3+6, ELHUM)
AM3 = ELLYFO(ICHT3+7, ELHUM)
AH3 = ELLYFO(ICHT3+8, ELHUM)
ICOUH(IHTGPH, ILAYER) = ICOUH(IHTGPH, ILAYER) + 1
IPHT = ICOUH(IHTGPH, ILAYER)
TENSTF(IHTGPH, 3+IPHT, ILAYER) = IPHT
TENSTF(IHTGPH, 12+IPHT, ILAYER) = EPSY
TENSTF(IHTGPH, 66+IPHT, ILAYER) = ILY
TENSTF(IHTGPH, 60+IPHT, ILAYER) = ARATIO
TENSTF(IHTGPH, 63+IPHT, ILAYER) = SIGYLD
INT = 9*(IPHT-1) + 1
TENSTF(IHTGPH, 30+INT, ILAYER) = AL1
TENSTF(IHTGPH, 31+INT, ILAYER) = AM1
TENSTF(IHTGPH, 32+INT, ILAYER) = AR1
TENSTF(IHTGPH, 34+INT, ILAYER) = AM2
TENSTF(IHTGPH, 36+INT, ILAYER) = AL2
TENSTF(IHTGPH, 37+INT, ILAYER) = AM3
TENSTF(IHTGPH, 38+INT, ILAYER) = AH3
ENDIF
ELSE
EPSY = EPSY1
ILY = ILY1
SIGYLD = SIGYD1
ARATIO = ARATOl
ICHT3 = ICHT1 + 9
AL1 = ELLYFO(ICHT3, ELHUM)
AM1 = ELLYFO(ICHT3+1, ELHUM)
AR1 = ELLYFO(ICHT3+2, ELHUM)
AL2 = ELLYFO(ICHT3+3, ELHUM)
AM2 = ELLYFO(ICHT3+4, ELHUM)
AR2 = ELLYFO(ICHT3+5, ELHUM)
AL3 = ELLYFO(ICHT3+6, ELHUM)
AM3 = ELLYFO(ICHT3+7, ELHUM)
AH3 = ELLYFO(ICHT3+8, ELHUM)
ICOUH(IHTGPH, ILAYER) = ICOUH(IHTGPH, ILAYER) + 1
IPHT = ICOUH(IHTGPH, ILAYER)
TENSTF(IHTGPH, 3+IPHT, ILAYER) = IPHT
TENSTF(IHTGPH, 12+IPHT, ILAYER) = EPSY
TENSTF(IHTGPH, 66+IPHT, ILAYER) = ILY
TENSTF(IHTGPH, 60+IPHT, ILAYER) = ARATIO
TENSTF(IHTGPH, 63+IPHT, ILAYER) = SIGYLD
INT = 9*(IPHT-1) + 1
TENSTF(IHTGPH, 30+INT, ILAYER) = AL1
TEHSTF(IHTGPH,31+INT,ILAYER) = AM1
TEHSTF(IHTGPH,32+INT,ILAYER) = AS1
TEHSTF(IHTGPH,33+INT,ILAYER) = AL2
TEHSTF(IHTGPH,34+INT,ILAYER) = AM2
TEHSTF(IHTGPH,35+INT,ILAYER) = AH2
TEHSTF(IHTGPH,36+INT,ILAYER) = AL3
TEHSTF(IHTGPH,37+INT,ILAYER) = AM3
TEHSTF(IHTGPH,38+INT,ILAYER) = AH3

ENDIF
ENDIF
ENDIF
ENDIF
RETURN
END

C __________________________________________ SUBC0W ________________ C
C
C INCLUDE(PROCESS)
SUBROUTINE SUBC0W(PDIR,SIGMA1,SIGMA2,SIGMA3,PROPER,L,J,COHVER,
$ IMPLICIT REAL*8(A-H,0-Z)
C . . .SWITCHES; REHUHB=100:10,FORMAT=900:10
C . . .SWITCHES:
C*******************************************************************************
C C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C STATUS
C C*******************************************************************************

COMMOH /BLOCKB/HISTO(27,70,1S) ,CRACK(27,2B0,1B) ,IHISTY,IHIST1
$,IHIST2
COMMOH /ABC/TEHSTF(27,80,1S),ITHSPG,ITHSG1,ITHSG2
DIMENSION PROPER(25),PDIR(3,3)

C

SUBROUTINE TO CHECK THE GAUSS POINT 'L' OF CONCRETE LAYER 'J' FOR THE CURRENT ELEMENT. THIS CHECK IS DONE FOR A POINT WHICH WAS PREVIOUSLY UNLOADING OR LOADING AND IN THE CURRENT ITERATION IS DETERMINED AS BEING IN A LOADING CONDITIONS (INCREASE IN STRAINS OF AT LEAST ONE DIRECTION) OR FOR A POINT SHOWING STRAINS WITH DIFFERENT SIGNS IN THE CURRENT AND PREVIOUS ITERATION.

C

HIST0(L,10,J) = PDIR(1,1)
HIST0(L,11,J) = PDIR(1,2)
HIST0(L,12,J) = PDIR(1,3)
HIST0(L,13,J) = PDIR(2,1)
HIST0(L,14,J) = PDIR(2,2)
HIST0(L,15,J) = PDIR(2,3)
HIST0(L,16,J) = PDIR(3,1)
HIST0(L,17,J) = PDIR(3,2)
HIST0(L,18,J) = PDIR(3,3)

C

DETERMINE STATE OF STRESS AND RATIO OF PRIBC. STRESSES

C
CALL STATUS (SIGMA1, SIGMA2, SIGMA3, STATE, J, PROPER, L, IF0C)
HIST0(L,28,J) = STATE
100 CONTINUE
IF (IFLAG1.EQ.1 .OR. IFLAG2.EQ.1 .OR. IFLAG3.EQ.1) THEN

C
C STRAINING IN THE OPPOSITE DIRECTION ....

IF (IFLAG1 .EQ. 1) THEN
   LET THE CURRENT STRAIN REBOUND UP TO ZERO VALUE
   FOR THE TIME BEING.
   HISTO(L,31,J) = 0.0
   HISTO(L,22,J) = 0.0
ENDIF

IF (IFLAG2 .EQ. 1) THEN
   HISTO(L,32,J) = 0.0
   HISTO(L,23,J) = 0.0
ENDIF

IF (IFLAG3 .EQ. 1) THEN
   HISTO(L,33,J) = 0.0
   HISTO(L,24,J) = 0.0
ENDIF

C SET THE STATE OF THE POINT TO LOADING AND CARRY OUT
C FURTHER ITERATIONS.

   HISTO(L,30,J) = 1.
   CONVER = 999.0
RETURN
ENDIF

C THE POINT IS RELOADING. ITS SECANT 'E' WILL REMAIN
C UNCHANGED UP TO THE STRAIN FROM WHICH UNLOADING
C INITIATED. FROM THERE ON THE VIRGIN STRESS-STRAIN
C CURVE WILL BE FOLLOWED.

IF (DABS(EPS1) .GE. .999*DABS(HISTO(L,31,J))) THEN
   LET THE CURRENT STRAIN BE EQUAL TO THE STRAIN AT THE
   INITIATION OF UNLOADING AND CARRY OUT FURTHER ITERATION
   HISTO(L,22,J) = HISTO(L,31,J)
   HISTO(L,30,J) = 1.
   CONVER = 999.
ENDIF

IF (DABS(EPS2) .GE. .999*DABS(HISTO(L,32,J))) THEN
   HISTO(L,23,J) = HISTO(L,32,J)
   HISTO(L,30,J) = 1.
   CONVER = 999.
ENDIF

IF (DABS(EPS3) .GE. .999*DABS(HISTO(L,33,J))) THEN
   HISTO(L,24,J) = HISTO(L,33,J)
   HISTO(L,30,J) = 1.
   CONVER = 999.
ENDIF
RETURN
END
SUBROUTINE CDB12(SIGMA1, SIGMA2, SIGMA3, L, J, EPS1, EPS2, EPS3, NINC, $ NIT, PROPER, ELMNUM, IOUT, CONVER, BSTIFF, TOLER, PDIR, CHKUS, ITYPE $, DSTIF, IFG, ERROR, IFLG1, IFLG2, IFLG3, ITCK, ICOUNT, IFLAGS)

IMPLICIT REAL*8(A-H, O-Z)

C...SWITCHES: REHUHB=100:10, FÜRMAT=900:10
C...SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C STATUS FAILURE FINDEX MATERL PRECON CHKUS
C
INTEGER ELNUR
COMMON /BL0CKS/HIST0(27,70,15), CRACK(27,260,15), HIISTY, HIIST2
$ , HIIST3
COMMON /ABC/TENSTF(27,80,15), ITNSG, ITNSG1, ITNSG2
COMMON/TRANS/V1(3), V2(3), V3(3)
DIMENSION PROPER(25)
REAL*8 BUD, PDIR(3,3), DSTIF(6,6)
LOGICAL CHKUS

C SUBROUTINE TO CHECK CONCRETE MATERIAL LAW IN THE
C PRE-ULTIMATE PHASE.
C PREVENT OVERSHOOTING WHEN THE STRESS POINT IS IN THE
C PRE-ULTIMATE STAGE AND LOADING.
C
C ...... TOLERE = VERY SMALL INTERNALLY SET TOLERANCE FOR
C MODULUS OF ELASTICITY WHEN OVERSHOOTING
C IS TAKING PLACE.
C ...... TOLERN = VERY SMALL, INTERNALLY SET, TOLERANCE
C FOR POISSON'S RATIO WHEN THERE IS
C OVERSHOOTING.
C
C TOLER = TOLER
TOLERN = TOLER
C
C OVRSHT = 0.0 ...... THE STRESS POINT HAS NOT REACHED ITS
C ULTIMATE VALUE WITH THE CURRENT RATIO
C
C OVRSHT = 1.0 ...... THE STRESS POINT HAS JUST REACHED OR
C EXCEEDED ITS ULTIMATE VALUE.
C
C INITIALIZE FUNCT,OVRSHT
C
C C1 = 0.0
C C2 = 0.0
C C3 = 0.0
C FUNCT = 0.0
C STATE = 0.0
C OVRSHT = 0.0
C BS = 0.0
C BUS = 0.0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CALCULATE THE RATIO OF MAX. TO MIN. PRINCIPAL
STRESS AND THE STATE OF STRESS.

\[
\pi = 3.141692654
\]

FC = PROPER(6)
FT = PROPER(6)
CALL STATUS (SIGMA1, SIGMA2, SIGMA3, STATE, J, PROPER, L, IFC)
IF (CRACK(J,L,J) .GE. 1.0) THEN
  IF (IFC .EQ. 2) THEN
    IF (HIST0(L,41,J) .LT. PROPER(7)) THEN
      FC = FC*(1.0-0.2*(HIST0(L,41,J)/PROPER(7))
    ELSE
      FC = FC*0.8
    ENDIF
    FC = DMIN1(HIST0(L,68,J),FC)
  ENDIF
ENDIF
ENDIF

SAVE CURRENT STATE OF TRIAXIAL STRESS

HIST0(L,28,J) = STATE
IF (STATE.EQ.100. .OR. STATE.EQ.110. .OR. STATE.EQ.111) THEN
  STATE OF UNIAXIAL OR BI/TRI AXIAL TENSION.
  IF (SIGMA1 .GE. FT) THEN
    THE STRESS POINT HAS JUST REACHED OR VIOLATED THE
    FAILURE ENVELOPE. TO AVOID OVERSHOOTING FORCE THE
    POINT TO BE ON THE FAILURE ENVELOPE.
    OVRSHT = 1.0
    HIST0(L,29,J) = 2.
  ENDIF
ENDIF

NOW CALCULATE SECANT VALUES AT FAILURE USING THE RATIO
OF PRINCIPAL STRESSES DETERMINED AT THE PREVIOUS ITER-
ATION.

CALL FAILUR (L, J, PROPER, IOUT, ES, MUS, SIGMA1, SIGMA2,
1 SIGMA3, IFC, ELNUM)
GO TO 160
ENDIF
GO TO 160

STATE OF TENSION-COMPRESSION (CRACKING ZONE)

IF (STATE .EQ. 101.) CALL FINDEX (FUNCT, FC, PROPER, SIGMA1,
1 SIGMA2, SIGMA3, IOUT)

STATE OF TENSION-COMPRESSION (CRUSHING ZONE)

IF (STATE .EQ. -101.) CALL FINDEX (FUNCT, FC, PROPER, SIGMA1,
1 SIGMA2, SIGMA3, IOUT)

STATE OF UNIAXIAL, BIAXIAL OR TRIAXIAL COMPRESSION
IF (STATE.EQ.(-100.) .OR. STATE.EQ.(-110.) .OR. STATE.EQ.(-111.))
1 CALL FINDEX (Funct, FC, PROPER, SIGMA1, SIGMA2, SIGMA3, IOUT)
IF (STATE.EQ.100. .AND. STATE.EQ.110. .AND. STATE.EQ.111.) HISTO(L 1 ,J) = FUNCT
IF (Funct .GE. (-.01)) THEN

THE STRESS POINT HAS JUST REACHED OR VIOLATED THE
FAILURE ENVELOPE. FORCE THE STRESS POINT TO BE ON
THE FAILURE SURFACE FOR THE CURRENT ITERATION.

UVRSHT = 1.0
HISTO(L,29,J) = 2.

CALCULATE THE SECANT VALUES AT FAILURE FOR THE POINT.

CALL FAILUR (L, J, PROPER, IOUT, ES, HUS, SIGMA1, SIGMA2, SIGMA3, IPC, ELMN)

THERE IS A POSSIBILITY THAT OVERSHOOTING AROUND THE
ULTIMATE POINT IS TAKING PLACE. COMPARE THE CURRENT
PRINCIPAL STRESSES WITH THE CALCULATED VALUES AT
ULTIMATE. IF THE DIFFERENCE IS WITHIN 2% THEN USE A
VERY SMALL TOLERANCE OF 2.0% FOR E TO AVOID EXCESSIVE
SOFTENING THAT WILL CAUSE CONVERGENCE PROBLEMS. THE
SELECTED TOLERANCE WILL ENSURE THAT CONVERGENCE IS
ACHIEVED FROM THE STIFF SIDE. THIS TOLERANCE WILL BE
IF EFFECT FOR THE CURRENT LOAD INCREMENT.

ALSO SET THE TOLERANCE FOR POISSON'S RATIO TO 2.0%
THIS TOLERANCE EFFECTS THE ADJUSTED VALUE DETERMINED
FOR THE MODULUS OF ELASTICITY IN SUB. 'PRECOB'.

C1 = 3.
C2 = 3.
C3 = 3.
SIGMA1 = HISTO(L,34,J)
SIGMA2 = HISTO(L,35,J)
SIGMA3 = HISTO(L,36,J)
IF (STATE .EQ. 100) GO TO 130
IF (STATE .EQ. 110) GO TO 110
IF (DABS(SIGMA3) .EQ. 0.0) GO TO 100
C3 = 100.*((DABS(SIGMA1)-DABS(SIGMA3))/DABS(SIGMA3))
100 CONTINUE
IF (STATE .EQ. (-100)) GO TO 140
110 CONTINUE
IF (DABS(SIGMA2) .EQ. 0.0) GO TO 120
C2 = 100.*(DABS(SIGMA2)-DABS(SIGMA3))/DABS(SIGMA2)
120 CONTINUE
IF (STATE .EQ. (-110)) GO TO 140
130 CONTINUE
IF (DABS(SIGMA1) .EQ. 0.0) GO TO 140
C1 = 100.*(DABS(SIGMA1)-DABS(SIGMA3))/DABS(SIGMA3)
140 CONTINUE
IF (C1.LE.2. .OR. C2.LE.2. .OR. C3.LE.2.) THEN
  TOLER = 2.0
  TOLERH = 2.0
ENDIF
GO TO 150
CALL THE MATERIAL SUBROUTINE TO DETERMINE CURRENT
VALUES OF SECANT 'E' AND 'NU'.

CALL MATERIAL (L, J, SIGMA1, SIGMA2, SIGMA3, STATE, ES, NUS, IDOUT,
              1  PROPER, BETHA, IPC, ELNUM)

CONTINUE
HISTO(L,10,J) = PDIR(1,1)
HISTO(L,11,J) = PDIR(1,2)
HISTO(L,12,J) = PDIR(1,3)
HISTO(L,13,J) = PDIR(2,1)
HISTO(L,14,J) = PDIR(2,2)
HISTO(L,16,J) = PDIR(3,1)
HISTO(L,17,J) = PDIR(3,2)
HISTO(L,18,J) = PDIR(3,3)
IF (NINO .EQ. 1) THEN
  FOR THE FIRST LOAD STEP THE STRESS POINT MUST BE
  INSIDE THE FAILURE ENVELOPE.
  IF (STATE.EQ.100. .OR. STATE.EQ.110. .OR. STATE.EQ.111.) THEN
    IF (SIGMA1 .GT. FT) THEN
      WRITE (IDOUT, 900) SIGMA1, FT
      WRITE (IDOUT, 910) L, J, ELNUM
      STOP
    ENDIF
  ENDIF
  IF (FUNCT .GT. 0.0) THEN
    WRITE (IDOUT, 900) SIGHA1, FT
    WRITE (IDOUT, 910) L, J, ELNUM
    STOP
  ENDIF
ENDIF

CHECK CONVERGENCE FOR E AND NU
CALL PRECON (ES, NUS, CONVER, HSTIFF, TOLER, TOLER, CHKGUS, L, J
              1  , IDOUT, OVRSHT, ELNUM)

IF (HISTO(L,29,J) .EQ. 2.) THEN
  IF (STATE.EQ.100. .OR. STATE.EQ.110. .OR. STATE.EQ.111 .OR.
      1  STATE.EQ.101) THEN
    POSSIBILITY OF A NEW CRACK
    IN ORDER TO HAVE A NEW CRACK THE ANGLE BETWEEN THE
    CURRENT PRINCIPAL DIRECTION, THETA, AND ANY EXISTING
    CRACK MUST BE LARGER THAN A THRESHOLD VALUE SPECIFIED
    EXTERNALLY. THIS IS DONE TO AVOID 'NUMERICAL CRACKING'.
    RECOVER THE THRESHOLD ANGLE 'ALFA'
    ALFA = PROPER(15)*3.1415926535/180.
    ALFA=60 = 3.1415926535/180.
A = DSIN(ALFA)

NCRACK = IABI(CRACK(L,1,J))

IF (NCRACK .GE. 1) THEN

IF THE ANGLE BETWEEN CURRENT PRINCIPAL DIRECTION AND AN
NORMAL TO THE EXISTING CRACKS IS LESS THAN 'ALFA' DO NO
INITIATE A NEW CRACK.

AL1 = HISTO(L,10,J)
AMI = HISTO(L,11,J)
AMI1 = HISTO(L,12,J)
DO 170 ICR = 1, NCRACK

IMP = 9*(ICR-1) + 1
ACL = CRACK(L,137+IMP,J)
ACM = CRACK(L,138+IMP,J)
ACH = CRACK(L,139+IMP,J)

DOT = AL1*ACL + AMI*ACM + AMI1*ACH
IF (DOT .GT. 1.0) DOT = 1.0
IF (DOT .LT. (-1.0)) DOT = -1.0
B = DSQRT(1.0000-DOT*DOT)
IF (B .LT. A) THEN

... NO NEW CRACK

HISTO(L,29,J) = 1.0
RETURN
ENDIF
170 CONTINUE
ENDIF
IF (IFLAG 8 .EQ. 4) THEN

D0T1=V3(1)*PDIR(1,1)+V3(2)*PDIR(1,2)+V3(3)*PDIR(1,3)
IF (DABS(DOT1) .GE. 0.99) RETURN
ENDIF

INITIATION OF ANOTHER CRACK. STORE THE INFORMATION

CALL CKIHIT (L, J, PROPER, ELBUH, NEL, IOUT, 1TYPE, 1
DSTIF, IERROR, IFLAG1, IFLAG2, IFLAG3, ITCRK, 1COUP,
2 IFLAG8)

IF (STATE .EQ. 110.) THEN

GET RATIO OF MAX. TO MIN. PRINCIPAL TENSILE STRESSES

RATIO = SIGMA2/SIGMA1

FOR EQUAL BIAXIAL TENSION TWO ORTHOGONAL CRACKS CAN
INITIATE AT THE SAME TIME. CHANGE

IF (RATIO.GE.0.99999 .AND. RATIO.LT.1.00001) THEN

AL1 = PDIR(2,1)
AMI = PDIR(2,2)
AMI1 = PDIR(2,3)
DO 180 ICR = 1, NCRACK

IMP = 9*(ICR-1) + 1
ACL = CRACK(L,137+IMP,J)
ACM = CRACK(L,138+IMP,J)
ACN = CRACK(L,139+IMP,J)

DOT = AL1*ACL + AMI*ACM + AMI1*ACN
IF (DOT .GT. 1.0) DOT = 1.0
IF (DOT .LT. (-1.0)) DOT = -1.0
B = DSQRT(1.0000-DOT*DOT)
IF (B .LT. A) THEN

... NO NEW CRACK

HISTO(L,29,J) = 1.0
RETURN
ENDIF
180 CONTINUE
ENDIF

ENDIF
ACM = CRACK(L, 139 + IMP, J)
ACN = CRACK(L, 138 + IMP, J)
DOT = AL1*ACL + AM1*ACM + AH1*ACN
IF (DOT .GT. 1.0) DOT = 1.0
IF (DOT .LT. (-1.0)) DOT = -1.0
B = DSQRT(1.0DDD - DOT*DOT)
IF (B .LT. A) THEN

... NO NEW CRACK

HISTO(1, 29, J) = 1.0
GO TO 190
ENDIF

190 CONTINUE

IF (STATE .EQ. 111.) THEN

GET RATIO OF MAX. TO MIN. PRINCIPAL TENSILE STRESSES

RATIO = SIGMA3/SIGMA1

FOR EQUAL TETRAXIAL TENSION THREE ORTHOGONAL CRACKS CAN
INITIATE AT THE SAME TIME. CHANGE

IF (RATIO.GE.0.99999 .AND. RATIO.LT.1.00001) THEN

AL1 = PDIR(3,1)
AM1 = PDIR(3,2)
AH1 = PDIR(3,3)
DO 200 ICR = 1, NCRACK
IMP = 9*(ICR-1) + 1
ACL = CRACK(L, 137 + IMP, J)

ACM = CRACK(L, 138 + IMP, J)
ACN = CRACK(L, 139 + IMP, J)
DOT = AL1*ACL + AM1*ACM + AH1*ACN
IF (DOT .GT. 1.0) DOT = 1.0
IF (DOT .LT. (-1.0)) DOT = -1.0
B = DSQRT(1.0DDD - DOT*DOT)
IF (B .LT. A) THEN

...
... SO NEW CRACK

    HISTO(L,29,J) = 1.0
    GO TO 210
ENDIF

CONTINUE
IF (IFLAGS .EQ. 4) THEN
    DOT3 = V3(1)*PDIR(3,1) + V3(2)*PDIR(3,2) + V3(3)*PDIR(3,3)
    IF (DABS(DOT3) .GE. 0.99) GO TO 210
ENDIF

ROTATE INTACT CONCRETE'S PRINCIPAL AXES TO THE OTHER
PRINCIPAL DIRECTION AND FORM A CRACK ORTHOGONAL TO IT.

    HISTO(L,10,J) = PDIR(3,1)
    HISTO(L,11,J) = PDIR(3,2)
    HISTO(L,12,J) = PDIR(3,3)
    HISTO(L,13,J) = PDIR(1,1)
    HISTO(L,14,J) = PDIR(1,2)
    HISTO(L,15,J) = PDIR(1,3)
    HISTO(L,16,J) = PDIR(2,1)
    HISTO(L,17,J) = PDIR(2,2)
    HISTO(L,18,J) = PDIR(2,3)

CALL CHKEXIT (L, I, PROPER, ELNUM, BUEL, IOUT,
1       ITYPE, DISTIF, TERROR, IFLAG1, IFLAG2, IFLAG3,
2       ITRK, ICOT, IFLAG8)

ENDIF

INTACT CONCRETE IN PRECRACKING PHASE AGAIN.

210 CONTINUE
    HISTO(L,29,J) = 1.0

SET INTACT CONCRETE'S MODULI TO THOSE CALCULATED AT
THE OFFSET OF CRACKING.

    HISTO(L,20,J) = ES
    HISTO(L,21,J) = HUS

RECOVER THE STRESSES AT THE INITIATION OF CRACKING AND
CALCULATE THE STRAINS AT CRACKING; THESE STRAINS ARE
THE STRAINS AT THE INITIATION OF UNLOADING FOR INTACT
CONCRETE.

    SIGM1F = HISTO(L,34,J)
    SIGM2F = HISTO(L,35,J)
    SIGM3F = HISTO(L,36,J)

    EPS1 = (SIGM1F-HUS*SIGM2F-HUS*SIGM3F)/ES
    EPS2 = (SIGM2F-HUS*SIGM1F-HUS*SIGM3F)/ES
    EPS3 = (SIGM3F-HUS*SIGM2F-HUS*SIGM1F)/ES
    AEG = DMAX1(DMAX1(DABS(EPS1),DABS(EPS2)),DABS(EPS3))
    IF (DABS(EPS1) .LT. AEG*1E-3) EPS1 = 0.0
    IF (DABS(EPS2) .LT. AEG*1E-3) EPS2 = 0.0
    IF (DABS(EPS3) .LT. AEG*1E-3) EPS3 = 0.0

    HISTO(L,31,J) = EPS1
HIST0(L,32,J) = EPS2
HIST0(L,33,J) = EPS3

THE CONSTITUTIVE MATRIX FOR THE POINT MUST BE REGENERATE
CHEGS = .TRUE.

STIFFNESS MATRIX FOR THE ELEMENT MUST BE RECALCULATED.
NSTIFF = 1

ENDIF

EVEN THOUGH THE MATERIAL LAW IS SATISFIED AROUND THE
ULTIMATE POINT THE EQUILIBRIUM MAY NOT BE. THIS IS DUE
TO THE FACT THAT WE FORCED THE STRESS POINT THAT WAS
OVERSHOOTING TO LIE ON THE FAILURE ENVELOPE FOR THE
TIME BEING. FURTHER ITERATIONS ARE NECESSARY TO LOCATE
THE STRESS POINT EXACTLY.

CONVER = 999.

ENDIF

RETURN

900 FORMAT(1X,'SIGMA=',D12.5,'FT=',D12.5)
910 FORMAT(/,5X,' ERROR',/,10X,'FOR THE FIRST LOAD STEP'
1 , '/THE STRESS POINT AT GAUSS POINT',12,IX,'OF LAYER'
2 , '/J2,IX,'LIES OUTSIDE THE FAILURE ENVELOPE FOR ELEMENT'
3 , '/.10X,NO.',12,'IN SUB CON12')

END

-------------------------------------------------------------------------------
SUBROUTINE PRECON (ES,NUS,CONVER,NSTIFF,TOLER, TOLERE,
$ CHKGUS, L , J ,IOUT,OVRSHT. lELEM)
IMPLICIT REAL*8(A-H,0-Z)

C...SWITCHES: REHUB=100:10 FORMAT=900:10
C...SWITCHES:
C INCLUDE(PROCESS)
SUBROUTINE PRECON(ES,NUS,CONVER,NSTIFF,TOLER, TOLERE,
$ CHKGUS,L,J,IOUT,OVRSHT,IELEM)
$ IMPLICIT REAL*8(A-H,0-Z)
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
COMMON /BLOCKB/HISTO(27,70,15),CRACK(27,250,15),IHISTY,IHISTI,$
$ ,IHIST2
COMMON /ABC/TENS(T,27,80,16),ITHSPUT,ITHST1,ITHSTI
REAL*8 BUS,BUO,BUW,LOWD,LOWBD
LOGICAL CHEGS

SUBROUTINE TO CHECK CONVERGENCE OF THE CONCRETE
CONSTITUTIVE LAW WHEN THE STRESS POINT IS IN THE
PRE-FAILURE STAGE.

COMPARE % DIFFERENCE FROM THE PREVIOUS VALUE.
\[ Cl = 100 \times (\text{HISTO}(L,20,J) - ES)/ES \]
\[ C2 = 100 \times (\text{NUS} - \text{HISTO}(L,21,J))/\text{NUS} \]

**IF** (\( C1 > \text{TOLER} \) AND \( C2 < \text{TOLER} \)) **THEN**

- For faster convergence, use the min. of the two 'E' values; one determined as the average of the current and updated secant E and the other determined by reducing the current E by 'TOLER' percent.

**IF** (\( \text{OVRSHT} = 0.0 \)) **THEN**

\[ \text{EAVERG} = (\text{HISTO}(L,20,J) + ES)/2. \]
\[ \text{EREDUC} = (1 - \text{TOLER}/100) \times \text{HISTO}(L,20,J) \]
\[ \text{HISTO}(L,20,J) = \text{DMIN}1(\text{EAVERG}, \text{EREDUC}) \]

**ENDIF**

Possibility of overshooting. Reduce 'E' gradually.

**IF** (\( \text{OVRSHT} \neq 0.0 \)) **THEN**

\[ \text{EAVERG} = \frac{(\text{HISTO}(L,20,J) + ES)}{2}. \]
\[ \text{EREDUC} = (1 - \text{TOLER}/100) \times \text{HISTO}(L,20,J) \]

**ENDIF**

If the compared 'ES' was the value at ultimate the stress point is still in the pre-ultimate phase.

\[ \text{HISTO}(L,20,J) = 1. \]

Update the constitutive law at this point.

\[ \text{CHAGUS} = .\text{TRUE}. \]

Regenerate stiffness matrix for the current element.

\[ \text{NSTIFF} = 1 \]

Carry out further iterations.

**CONVER** = 999.

Return.

**ENDIF**

**IF** (\( C2 > \text{TOLER} \)) **THEN**

- In this case, the Poisson's ratio is determined to be changed. However, to insure an overall softening of the constitutive matrix, when the Poisson's ratio is increased, the Young's modulus should also be decreased. This decrease is accomplished as follows:
  
  (Explanation in my notes)

- This is important in biaxial compression case where increase in '\( \nu \)' could decrease the compressive stress and result in false detection of unloading in subsequent iterations.

\[ \text{NUOLD} = \text{HISTO}(L,21,J) \]
\[ \text{EOLD} = \text{HISTO}(L,20,J) \]
\[ \text{NUNEW} = (1 + \text{TOLER}/100) \times \text{NUOLD} \]

Determine the adjusted Young's modulus corresponding.
TO 'HUEW' TO ENSURE AN OVERALL SOFTENING OF THE
CONSTITUTIVE RELATIONSHIP.

EADJST = EOLD*{(1.+TOLER/100.)*2.+POOLD**2.}/((1.+TOLER/100.)*1.)

MAKE SURE THAT 'EADJST' DOES NOT DECREASE BELOW 'ES'.

IF (EADJST .GE. ES) GO TO 100
IF (EADJST .LT. ES) THEN

THE REQ'D. 'E' CORRESPONDING TO 'HUEW' BECOMES LESS
THAN ITS LIMITING CASE IN THIS ITERATION. SET 'FIBAL'
TO ITS LIMITING CASE 'ES' AND THEN CALCULATE BACK ITS
CORRESPONDING POISSON'S RATIO TO ENSURE AN OVERALL
SOFTENING AND NOT EXCEEDING THE CURRENT REQ'D. LIMITS.

EADJST = ES

DETERMINE THE UPPER AND LOWER BOUNDS FOR 'HUEW'.

UPBBD = DSQRT(1.-ES*(1.-POOLD**2.)/EOLD)
CONST = EOLD*POOLD/((1.-POOLD**2.)*ES)
LOWBBD = -(1.0-DSQRT(1.+4.00*CQHST*CQHST))/(2.0*CQHST)

WRITE('*,*) 'LOWBBD OF EU',LOWBBD,'UPBBD OF EU',UPBBD

CHECK FOR ANY ERRORS.

IF (UPBBD .LT. .99*LOWBBD) THEN
  WRITE (IOUT, 900) L, J, UPBBD, LOWBBD, IELEM, ES, EOLD
  DIFF = DABS(100.*(EQLD-ES)/ES)
  IF (DIFF .LE. 5.) RETURN
endif
ENDIF

IF (HUSEW .LE. UPBBD .AND. HUSEW .GE. LOWBBD) THEN
  HUEW = HUSEW
  GO TO 100
ENDIF

IF (HUSEW .LE. UPBBD .AND. HUSEW .GE. LOWBBD) GO TO 100

FINALLY WHEN NONE OF THE ABOVE CONDITIONS ARE
SATISFIED ARBITRARILY SELECT 'HUEW'.

HUEW = UPBBD
IF (HISTO(L,29,J) .EQ. 2.) GO TO 110

ENDIF

100 CONTINUE
HISTO(L,29,J) = 1.

110 CONTINUE
HISTO(L,20,J) = EADJST
HISTO(L,21,J) = HUEW

UPDATE CONSTITUTIVE LAW AT THIS POINT.

CHKGUS = .TRUE.
NSTIFF = 1
CONVER = 999.
ENDIF
RETURN
$900 FORMAT(///,1X,'ERROR IN SUB. PRECOD',//,5X,
1 'DURING CALCULATION OF THE POISSONS RATIO FOR',
2 'GAUSS POINT',//,12.1X,'OF LAYER',//,12.5X,
3 'THE UPPER BOUND BECAME LESS THAN THE LOWER',
4 'BOUND',//,5X,'UPPER BOUND',//,12.5X,
5 'LOWER BOUND',//,12.5X,'FOR THE CURRENT',
6 'ITERATION THE UTILIZED SECANT MODULUS BECAME',
7 'LESS THAN ITS UPDATED VALUE',//,5X,'FOR',
8 'DIFFERENCES LESS THAN % THE CONVERGENCE IS',
9 'ASSUMED AND THE PROCESS IS CONTINUED',
0 'ELBUN',//,12.5X,'ENEW',//,12.5X,'EOLD',//,12.5X)
C
END
C
C---------------------------------------------- 0 2 3 ----------------------------------------------
C
INCLUDE(PROCESS)
SUBROUTINE CON23(L,J,CONVER,HSTIFF,SIGMA1,_SIGMA2,_SIGMA3,PROPER,
$ IDOUT,TOLER,CHKGUS,IELEM,IFC)
IMPLICIT REAL*8(A-H,O-Z)
C . . SWITCHES: REHUMB=100:10,FORMAT=900:10
C . . SWITCHES:
C******************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C STATUS HUPOST LATPRP ENVLPE
C
C******************************************************************************

COMMON /BLOCK/HISTO(27,70,18),CRACK(27,2B0,1B) ,IHISTY,IHIST1,
$,IHIST2
COMMON /ABC/TEHSTF(27,80,18),ITHSPG, ITHSG1, ITHSG2
COMMON /OTTO/AC,BC,FK1,FK2,SIG1H,SIG2H,SIG3H,FC,AAA,BBB,FT,IFG
DIMENSION PROPER(26)
REAL*8 SUS,BU1,BUF,J2
LOGICAL CHKGUS
C
C
SUBROUTINE TO CHECK THE MATERIAL PARAMETERS IN THE
C POST-FAILURE STATE (FOR NON-CRACKED CONCRETE).
C
C
C . . DETERMINE RATIO AND STATE OF GENERAL STRESSES.
C
C
ES = 0.0
IFG = 0
EMH = 0.0
AC = 0.0
BC = 0.0
FK1 = 0.0
FK2 = 0.0
SIG1H = 0.0
SIG2H = 0.0
SIG3H = 0.0
FT = PROPER(6)
CALL STATUS(SIGMA1, SIGMA2, SIGMA3, STATE, J, PROPER, L, IFC)
HIST0(L,28,J) = STATE
IF (HIST0(L,29,J) .EQ. 2.) THEN
IF (CRACK(L,1,J) .GE. 1.) THEN
IF (STATE.EQ.100. .OR. STATE.EQ.110. .OR. STATE.EQ.111.
  .OR. STATE.EQ.112.) THEN

THE INTACT CONCRETE IS A CRACKED POINT IS IN THE STATE
OF EITHER UNI-, BI-, OR AXIAL TENSION, OR THE-COMP
CRACKING ZONE. THE POST FAILURE BEHAVIOR IS TAKEN
CARE OF THROUGH INTRODUCING CRACKS AND THE INTACT CONC.
IS ASSUMED TO BE IN THE PRE-CRACKING STAGE.

HIST0(L,29,J) = 1.
GO TO 100
ENDIF

IN PREVIOUS ITERATION THE STRESS POINT WAS ON THE
FAILURE ENVELOPE. SINCE THE STRESSES ARE INCREASING
THE POINT MUST NOW BE IN THE POST-FAILURE STAGE.
DECREASE THE PREVIOUS SECANT 'E' BY TOLER, AND
DETERMINE THE CORRESPONDING SECANT 'BU'...

HIST0(L,20,J) = (1.-TOLER/100.0)*HIST0(L,20,J)
CALL BUSPOST (L, J, HISTO(L,20,J),BUS,TOLER,TOUT)

NOW THE POINT IS IN THE POST-FAILURE SEGMENT...

HIST0(L,29,J) = 3.
HISTO(L,21,J) = BUS
UPDATE THE MATERIAL LAW AT THIS POINT.

CONTINUE
CHECUS = .TRUE.
HSTIFF = 1
CONVER = 999.
RETURN
ENDIF

RETRIEVE PRINCIPAL STRESSES AT FAILURE
SIGM1F = HISTO(L,34,J)
SIGM2F = HISTO(L,35,J)
SIGM3F = HISTO(L,36,J)

CHECK FOR OVERSHOOTING
IF (STATE .EQ. (-100.)) THEN
STATE OF UNIAXIAL COMPRESSION
IF (DABS(SIGMA3) .LE. DABS(SIGM3F)) GO TO 110
ENDIF
IF (DABS(SIGMA1).GT.DABS(SIGM1F) .OR. DABS(SIGMA2).GT.DABS(SIGM2F)
1 .OR. DABS(SIGMA3).GT.DABS(SIGM3F)) THEN

AT LEAST ONE PRINCIPAL STRESS HAS EXCEEDED ITS
CORRESPONDING VALUE AT ULTIMATE.

REGARDLESS OF THE CURRENT ALLOWABLE TOLERANCE REDUCE
THE CURRENT MODULUS VERY SLOWLY TO AVOID EXCESSIVE
SOFTENING AROUND THE ULTIMATE POINT. THIS WILL ELIMI-
NATE THE POSSIBILITY OF DIVERGENCE.

USE ARBITRARY DECREMENTS OF 1%

HISTO(L,20,J) = .990*HISTO(L,20,J)

CALL HPOST(L, J, HISTO(L,20,J), NUS, TOLER, IDOUT)
HISTO(L,21,J) = NUS

UPDATE THE MATERIAL LAW AT THIS POINT.
CHGUS = .TRUE.

NEW ELEMENT STIFFNESS MATRIX MUST BE GENERATED

HSTIFF = 1

FURTHER ITERATION MUST BE CARRIED OUT.
CONVER = 999.

RETURN
EHDIF

RETIRE THE LAYER PROPERTIES

110 CONTINUE
CALL LAYPRP(PROPER,J,FC,FT,SETHAA,EPSC,EI,NUI,NUF,DPOST,CSIGMA)

EC = SECANT MODULUS AT MAX. UNIAXIAL COMPRESSIVE STS.
A = RATIO OF INITIAL MODULUS TO 'EC'.

IF (CRACK(L,1,J) .GE. 1.0) THEN
IF (IFC .EQ. 2) THEN
IF (HISTO(L,41,J) .LT. PROPER(7)) THEN
FC = PROPER(8)*(1.0-0.2*(HISTO(L,41,J)/PROPER(7)))
EPSC = EPSC+((1.0+0.10*(HISTO(L,41,J)/PROPER(7)))
ELSE
FC = 0.8*FC
EPSC = 1.10*EPSC
ENDIF
ENDIF
ENDIF
ENDIF

FC = DMIN1(HISTO(L,58,J),FC)
HISTO(L,58,J) = FC

EC = FC/EPSC
A = EI/EC

DETERMINE FINAL VALUES OF THE PRINCIPAL STRESSES AT
THE TERMINATION OF THE DESCENDING PORTION OF THE
STRESS-STRAIN CURVE.

FISIG1 = CSIGMA*SIGM1F
FISIG2 = CSIGMA*SIGM2F
FISIG3 = CSIGMA*SIGM3F
Having 'SIGMA1', 'SIGMA2' and 'SIGMA3' determine the corresponding secant values in the post-failure stage.

IF (STATE .EQ. (-100.) .OR. STATE .EQ. (-110.) .OR. STATE .EQ. (-111.)) THEN

STATE of uniaxial or bi/tri-axial compression.

CHECK the possibility of crushing (complete loss in stiffness).

IF (STATE .EQ. (-100.)) THEN

IF (DABS(SIGMA3) .GE. DABS(FISIG3)) GO TO 120
ENDIF

IF (DABS(SIGMA1) .LT. DABS(FISIG1) .OR. DABS(SIGMA2) .LT. DABS(FISIG2) .OR. DABS(SIGMA3) .LT. DABS(FISIG3)) THEN

SET modulus of elasticity and the Poisson's ratio to very small numbers (eliminate stiffness)

HIST0(L,20,J) = PROPER(1)/1000.0
HIST0(L,21,J) = 0.001

SET the state of the point to crushing.

HIST0(L,29,J) = 4.

EVEN if there were some cracks at this point they can't exist after the point is crushed.

CRACK(L,1,J) = 0.0
CHKGUS = .TRUE.
NSTIFF = 1
CONVER = 999.
RETURN
ENDIF

DETERMINE the failure value of SIGMA3, 'SIGFL3', corresponding to SIGMA1 and SIGMA2

120 CONTINUE

IF (STATE .EQ. (-100.)) THEN

SIGMA1 = 0.0
SIGMA2 = 0.0
ENDIF

IFG = 1
AAA = -1000.0*FC
BBB = 0.0
CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGFL3, IOUT, STATE, SIGFL1, SIGFL2)

CALCULATE the second invariant of the deviatoric stresses corresponding to 'SIGMA1' & 'SIGMA2' SIGFL3

J2 = (SIGMA1*SIGMA1+SIGMA2*SIGMA2+SIGFL3*SIGFL3-SIGMA1*SIGMA2+SIGMA3*SIGMA3-SIGMA1*SIGMA2-SIGMA3*SIGMA3)/3.0D00
X = DSQRT(J2)/FC - 1.0D0/DSQRT(3.0D00)
EF = EC/(1.+4.*(A-1.)*X)

CALCULATION OF THE SECANT YOUNG'S MODULUS.

ES1 = EI/2. - BETHA*(EI/2.-EF)
ES2 = DSQRT((ES1+ES1)*(EF+EF)+BETHA*(DPOST*(1.-BETHA)-1.))
ES = ES1 - ES2

DETERMINE THE CORRESPONDING POISSON'S RATIO.

CALL HUPOST (L, J, ES, HUS, TOLER, IOUT)

ENDIF

IF (STATE .EQ. (-101.)) THEN

STATE OF COMPRESSION-TENSION (CRUSHING ZONE).

CHECK POSSIBILITY OF CRUSHING.

IF (DABS(SIGMA3) .LE. DABS(FISIG3)) THEN

ELIMINATE STIFFNESS OF THE CURRENT POINT.
SET THE ELASTIC MODULI TO SMALL NUMBERS TO
AVOID NUMERICAL PROBLEMS.

HISTO(L,20,J) = PROPER(1)/1000.0
HISTO(L,21,J) = 0.001

SET THE STATE OF THE POINT TO CRUSHING.

HISTO(L,29,J) = 4.

NO CRACKS CAN EXIST AT THIS POINT ANY MORE.

CRACK(L,1,J) = 0.0
CHEGUS = .TRUE.
NSTIFF = 1
CONVER = 999.
RETURN

ENDIF

IGF = 1
AAA = -1000.0*FC
BBB = 0.0

CALL ENVLPE (SIGMA1, SIGMA2, SIGMAS, PROPER, BETHA, SIGFL3, SIGFL2)

EF = EC

CALCULATION OF THE SECANT YOUNG'S MODULUS.

ES1 = EI/2. - BETHA*(EI/2.-EF)
ES2 = DSQRT((ES1+ES1)*(EF+EF)+BETHA*(DPOST*(1.-BETHA)-1.))
EMN = ES1 - ES2

RETRIEVE THE NECESSARY PARAMETERS THAT WERE CALCULATED
AND STORED AT THE ONSET OF FAILURE.

EA = HISTO(L,26,J)
EM = HISTO(L,56,J)
BETHAF = HISTO(L,55,J)
ES = (BETHA*EM+EA*EM)/(BETHA*EA+EM+BETHAF*EM+(EM-EM))

DETERMINE THE CORRESPONDING SECANT 'NU'.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CALL HUPOST (L, J, ES, HUS, TOLER, IOUT)

ENDIF

* CHECK CONVERGENCE *

DO NOT OVERSOFTEN:
IF THE UPDATED SECANT MODULUS IS SMALLER THAN ITS
CORRESPONDING VALUE IN THE CURRENT ITERATION THEN
USE THE CURRENT VALUE AND ASSUME CONVERGENCE

IF (ES .LE. HIST0(L,20,J)) THEN
WRITE (IOUT, 900) L, J, IHEM, ES, HISTO(L,20,J)
RETURN
ENDIF

COMPARE % DIFFERENCE BETWEEN THE VALUE USED IN THE
CURRENT ITERATION AND THE UPDATED VALUE OBTAINED
BASED ON

C = 100.*{(ES-HISTO(L,20,J))/ES
IF (C .GT. TOLER) THEN
HIST0(L,20,J) = (1.-TOLER/100.)*HIST0(L,20,J)
ENDIF

CALL HUPOST (L, J, HISTO(L,20,J), HUS,TOLER, IOUT)
HISTO(L,21,J) = HUS

UPDATE THE MATERIAL LAW.

CHEQUE = .TRUE.
NSTIFF = 1
CONVER = 999.
ENDIF
RETURN

900 FORMAT(/,2X, 'WARNING',/.,5X,'THE CALCULATED POST-',
1, 'FAILURE SECANT MODULUS AT G. POINT',/.,5X,'IN LAYER',
2, 'I=',II,'ELEM',/.,5X,'ES=',/.,12.5,'/.,5X,'WHICH IS LESS THAN ITS VALUE'
3, 'OF',/.,12.5,'/.,11X,'USED IN THE CURRENT ITERATION 3')

END

------------------------- HUPOST -------------------------

C INCLUDE(PROCESS)
SUBROUTINE HUPOST(L,J,ES,HUS,TOLER,IOUT)
IMPLICIT REAL*(A-H,O-Z)
C...SWITCHES: RENUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*************** HUPOST ****************

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SUBROUTINE TO DETERMINE THE POISSON'S RATIO IN THE POST-FAILURE STATE.

..... RETRIEVE THE SECANT YOUNG AND THE BULK MODULI AT FAILURE

ESFLR = HIST0(L,26,J)
BULKMD = HIST0(25,J)

FOR EACH 5% DECREASE IN 'ESFLR' THE POISSON'S RATIO IS INCREASED TO NUS=0.005*NUS IN WHICH:

BULKMD=E(POST-FAILURE)/3(1-2*NUS)

THIS HAS THE EFFECT OF INCREASING THE BULK MODULUS AT FAILURE GRADUALLY TO ACCOUNT FOR THE VOLUMETRIC INCREASE THAT TAKES PLACE IN THE POST-FAILURE PHASE.

C = (ESFLR-ES)/-6.05*ESFLR)
NUS = ES/(-6.*BULKMD) + .50
NUS = NUS+(1.+C)*.005)

LIMIT POISSON'S RATIO TO 0.45

IF (NUS .GT. 0.45) NUS = 0.45
RETURN
END

------------- U R L D -------------

SUBROUTINE UHRELD(EPS1, EPS2, EPS3, PROPER, CHKGUS, NSTIFF, CONVER, L , J )
IMPLICIT REAL*8(A-H,0-Z)

SWITCHES; REBUMB=100:10,FORHAT=900:10

SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
COMMON /BLOCKS/HIST0(27,70,15),CRACK(27,250,15),IHISTY,IHISTI $,IHIST2
COMMON /ABC/TESTF(27,80,15),ITBSFG,ITBSEG,ITBSEG2
DIMENSION PROPER(26)
LOGICAL CHKGUS
C
IF (DABS(EPS1).LT..999.|DABS(HISTO(L,22,J)) .OR. DABS(EPS2).LT..999 1 +DABS(HIST0(L,23,J)) .OR. DABS(EPS3).LT..999|DABS(HISTO(L,24,J) 2 )) THEN
HIST0(L,30,J) = 2.

CHECK TO SEE IF UNLOADING RESULTS IN COMPRESSIVE STRAINS BECOMING TENSILE STRAINS OR VICE VERSA.

VAR1 = EPS1*HIST0(L,22,J)
VAR2 = EPS2*HIST0(L,23,J)
VAR3 = EPS3*HIST0(L,24,J)

IF (VAR1.LT.0.0 .OR. VAR2.LT.0.0 .OR. VAR3.LT.0.0) THEN


IF (VAR1 .LT. 0.0) THEN
   HIST0(L,31,J) = 0.0
   HIST0(L,22,J) = 0.0
ENDIF

IF (VAR2 .LT. 0.0) THEN
   HIST0(L,32,J) = 0.0
   HIST0(L,23,J) = 0.0
ENDIF

IF (VAR3 .LT. 0.0) THEN
   HIST0(L,33,J) = 0.0
   HIST0(L,24,J) = 0.0
ENDIF

HISTO(L,30,J) = 1.0
HIST0(L,29,J) = 1.0

SET MODULUS OF ELASTICITY AND THE POISSON'S RATIO TO THEIR INITIAL VALUES.

HISTO(L,20,J) = PROPER(1)
HIST0(L,21,J) = PROPER(2)

GENERATE NEW CONSTITUTIVE MATRIX, REGENERATE NEW ELEMENT STIFFNESS MATRIX AND CARRY OUT ADDITIONAL ITERATIONS.

CHKGUS = .TRUE.
NSTIFF = 1
CONVER = 999.
RETURN
ENDIF
ENDIF
IF (DABS(EPS1) .GE. .999 + DABS(HISTO(L,22,J)) .OR. DABS(EPS2) .GE. .999 + DABS(HISTO(L,23,J)) .OR. DABS(EPS3) .GE. .999 + DABS(HISTO(L,24,J))) HISTO(L,30,J) = 3.

SAVE THE CURRENT STRAINS

HISTO(L,22,J) = EPS1
HISTO(L,23,J) = EPS2
HISTO(L,24,J) = EPS3
RETURN

END

//------------------------------------------------------------------------------
// ROCRAK
//------------------------------------------------------------------------------

INCLUDE(PROCESS)

SUBROUTINE ROCRAK(EPS1,SIGMA1,SIGMA2,SIGMA3,PROPER,L,J,ELNUM,HEMEL
$ ,CHEKGUS,NSTIFF,CONVER,JOUT,ITYPE,DTSTIF,ITERROR
$ ,IIGL01,IILG02,ILULONG,ITCRC,ICDUP,IILGLG)

IMPLICIT REAL*8(A-H,O-Z)

C...SWITCHES: REHUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*******************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C CKIHIT
C*******************************************************************************

INTEGER ELNUM
COMMON/TRANS/V1(3),V2(3),V3(3)
COMMON/BLOCK5/HISTO(27,70,16),CRACK(27,260,15),IHISTY,ITHIST1
$ ,ITHIST2
COMMON/ABC/TENSTF(27,80,16),ITBSPG,ITBSPG1,ITBSPG2

DIMENSION PROPER(26),DTSTF(6,6)

LOGICAL CHEKGUS

SUBROUTINE TO CHECK THE POSSIBILITY OF INITIATING A NEW CRACK IN AN ALREADY CRACKED POINT. EVEN THOUGH THE MAX. PRINCIPAL STRESS IS BELOW ITS CRACKING VALUE ACCORDING TO THE CURRENT STATE OF STRESS AND ITS PROXIMITY FROM THE FAILURE ENVELOPE, THE MAX. PRINCIPAL STRAIN CAN EXCEED ITS CORRESPONDING VALUE AT THE INITIATION OF THE FIRST CRACK LEADING TO A NEW CRACK.

EPSCRK = MAX. PRINCIPAL STRAIN OF INTACT CONCRETE AT INITIATION OF THE FIRST CRACK.

EPSCRK = HISTO(L,37,J)
IF (EPS1 .LT. EPSCRK) RETURN

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
THERE IS A POSSIBILITY OF A NEW CRACK FORMATION.

NCrack = INT(Crack(L,1,J))

ALFA = MIN. ALLOWABLE ANGLE BETWEEN ANY TWO CRACKS.

ALFA = PROPER(18)*3.1415926535/180.0
A = SIN(ALFA)

IF THE ANGLE BETWEEN CURRENT PRINCIPAL DIRECTION AND AN
NORMAL TO THE EXISTING CRACKS IS LESS THAT 'ALFA' DO NOT
INITIATE A NEW CRACK.

AL1 = HISTO(L,10,J)
AM1 = HISTO(L,11,J)
AH1 = HISTO(L,12,J)
AL2 = HISTO(L,13,J)
AM2 = HISTO(L,14,J)
AH2 = HISTO(L,15,J)
AL3 = HISTO(L,16,J)
AM3 = HISTO(L,17,J)
AH3 = HISTO(L,18,J)
DO 100 ICR = 1, NCrack
   IMP = 9*(ICR-1) + 1
   ACL = CRACK(L,137+IMP,J)
   ACM = CRACK(L,138+IMP,J)
   ACH = CRACK(L,139+IMP,J)
   DOT = AL1*ACL + AM1*ACM + AH1*ACH
   IF (DOT > 1.0) DOT = 1.0
   IF (DOT < -1.0) DOT = -1.0
   B = DSQRT(1.0-DOT^DOT)
   IF (B < A) RETURN

IF THE NEW CRACK IS FORMING AT ANGLES
GREATER THAN 45 (DEG.) FROM THE EXISTING ONES
THEN THIS MEANS THAT THE OTHER PRINCIPAL DIRECTION
IS REACHING THE CRACKING CONDITION. THIS NEW CRACK
SHOULD NOT, HOWEVER, BE DEVELOPED BASED ON THE STRAIN
RECORDED FOR THE PREVIOUS CRACKS FOR WHICH THE ROTATION
OF THE SAME PRINCIPAL DIRECTION COULD BE RESPONSIBLE.
THIS ALSO IMPLIES THAT THE ROTATION OF THE SAME PRIN-
CIPAL DIRECTION IS LIMITED TO 45 (DEG.) FROM ITS
PREVIOUS ORIENTATION IN THE LAST ITERATION.

IF (B > 0.50*DSQRT(2.000)) RETURN
100 CONTINUE
   IF (ZFLAGS .EQ. 4) THEN
      DOT3 = V3(1)*AL1 + V3(2)*AM1 + V3(3)*AH1
      IF (DABS(DOT3) .GE. 0.99) RETURN
   ENDIF

SAVE THE CURRENT PRINCIPAL STRESSES FOR INTACT CONCRETE
HISTO(L,34,J) = SIGMA1
HISTO(L,35,J) = SIGMA2
HISTO(L,36,J) = SIGMA3

INITIATE A NEW CRACK AND GENERATE THE RELATED DATA.
CALL CKINIT (L, J, PROPER, ELNUM, NHEL, IOUT, ITYPE, DTSTIF,
  1  TERROR, IFLAG1, IFLAG2, IFLAG3, ITCRK, ICOUP, IFLAGS)
C
C UPDATE THE CONSTITUTIVE LAW FOR THE POINT, UPDATE THE
C CORRESPONDING ELEMENT STIFFNESS MATRIX AND CARRY OUT
C FURTHER ITERATIONS.
C
C
CHKGUS = .TRUE.
HSTIFF = 1
COHVER = 999.
RETURN
END
C
C
C
C INCLUDE (PROCESS)
SUBROUTINE UPDMAT(MAXCRK,IOUT,PROPER,MAXCK2,L,J,DTSTIF,ICOUP,
  1  IFC,IPLG1,IFLG2,IFLG3,ITCRK,TEMP11,EPX1,EPY1,EPZ1,GAHAY1,GAMAY1
  2  $#GAMAZ1, IELE)
IMPLICIT REAL*8 (A-H,O-Z)
C.SWITCHES: HUMB=100:10,F0RMAT=900:10
C...SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DTRAHS DCOPMPS
COMMON /BL0CK5/HIST0(27,70,15),CRACK(27,250,15),IHISTY, IHISTl,
  1  $,IHIST2
COMMON /ABC/TENSTF(27,80,15), ITHSPG, ITHSG1, ITHSG2
DIMENSION PROPER(25),DCRACK(30,30),DCOH(6,6),DTSTIF(6,6)
REAL*8 BUS,P0R(3,3),P0R(6,6),TEMP11(6,6)
C
C SUBROUTINE TO UPDATE CONSTITUTIVE MATRIX IN THE GLOBAL
C AXES FOR THE CURRENT GAUSS POINT 'L' IN LAYER 'J'.
C ARRAY 'DCOH' CONTAINS THE CONSTITUTIVE LAW IN THE
C MATERIAL AXES AND WILL LATER RETURN THAT IN THE
C GLOBAL AXES.
C
C DO 110 II = 1, 6
   DO 100 JJ = 1, 6
      DCOH(II, JJ) = 0.0
      TEHXYZ(II, JJ) = 0.0
   100 CONTINUE
   110 CONTINUE
C
C EQ = HIST0(20, L)
C BUS = HIST0(21, L)
C
C UPDATE ELEMENTS OF THE 'D' MATRIX IN THE MATERIAL AXES
C
C CONST = ES/(ES+(1.0*BUS)*(1.0-2.0*BUS))
   DCOH(1,1) = CONST*(1.0-BUS)
   DCOH(2,1) = CONST*BUS
   DCOH(2,1) = DCOH(1,2)
DCOB(1,1) = DCOB(1,2)
DCOB(1,3) = DCOB(1,2)
DCOB(2,3) = DCOB(1,2)
DCOB(3,2) = DCOB(1,2)
DCOB(2,2) = DCOB(1,1)
DCOB(3,3) = DCOB(1,1)
DCOB(4,4) = ES/(2.0+(1.0+BUS))
DCOB(5,5) = DCOB(4,4)
DCOB(6,6) = DCOB(4,4)
HISTO(L,1,J) = DCOB(1,1)
HISTO(L,2,J) = DCOB(1,2)
HISTO(L,3,J) = DCOB(2,2)
HISTO(L,4,J) = DCOB(3,3)
HISTO(L,5,J) = DCOB(1,3)
HISTO(L,6,J) = DCOB(2,3)
HISTO(L,7,J) = DCOB(4,4)
HISTO(L,8,J) = DCOB(5,5)
HISTO(L,9,J) = DCOB(6,6)

C IF THE POINT IS CRUSHED STIFFNESS IS COMPLETELY LOST.
C REGISTER THE FULL ZERO CONSTITUTIVE MATRIX FOR THIS
C POINT.
C
IF (HISTO(L,29,J) .EQ. 4.) GO TO 190
C TRANSFER THE ABOVE TO 'DCOB' MATRIX.
C
C ROTATE 'DCOB' TO GLOBAL AXES.
C
PDIR(1,1) = HISTO(L,10,J)
PDIR(1,2) = HISTO(L,11,J)
PDIR(1,3) = HISTO(L,12,J)
PDIR(2,1) = HISTO(L,13,J)
PDIR(2,2) = HISTO(L,14,J)
PDIR(2,3) = HISTO(L,15,J)
PDIR(3,1) = HISTO(L,16,J)
PDIR(3,2) = HISTO(L,17,J)
PDIR(3,3) = HISTO(L,18,J)

CALL DTRANS (DCOB, PDIR, IGUT)
C
C IF THERE ARE CRACKS AT THIS POINT
C
IF (CRACK(L,1,J) .GE. 1.) THEN
CALL DCRACKS (DCOB, IGUT, MAXCRK, MAXCK2, DCRACK, L, J, IELEM)
C
IF (ICRUP .EQ. 0) THEN
IF (IFLG1.EQ.1 .AND. IFLG2.EQ.1 .AND. IFLG3.EQ.1) THEN
IMG = 0
DO 160 ISP = 1, 3
ISPG = TESTF(L,ISP,J)
IF (ISPG.GT.0 .AND. TESTF(L,ISP,J).EQ.1.0) THEN
ISMG = (ISPG-1)*9 + 1
IMG = IMG + 1
C
IF (IFG1.EQ.2) THEN
IF (IMG .EQ. 1) THEN
CRACK(L,87+ICRCK,J) = PROPER(1)/1000.0
C
... TENSION STIFFENING (STEEL) DIRECTION W.R.T TO 'GLOBAL'
C
DS(1,1) = TESTF(L,ISMG+90,J)
\begin{verbatim}
527
DB(1,2) = TEBSTF(L,ISMG+31,J)
DB(1,3) = TEBSTF(L,ISMG+32,J)
DB(2,1) = TEBSTF(L,ISMG+33,J)
DB(2,2) = TEBSTF(L,ISMG+34,J)
DB(2,3) = TEBSTF(L,ISMG+35,J)
DB(3,1) = TEBSTF(L,ISMG+36,J)
DB(3,2) = TEBSTF(L,ISMG+37,J)
DB(3,3) = TEBSTF(L,ISMG+38,J)
DO 130 IM = 1, 6
   DO 120 IH = 1, 6
      TEHSTZ(IM,IM) = 0.0
      120 CONTINUE
   CONTINUE

ISP2 = 0
ISP3 = 0
IF (ISPG .EQ. 1) ISP2 = 1
IF (ISPG .EQ. 2) ISP2 = 2
IF (ISPG .EQ. 3) ISP2 = 3
ELSE IF (ISPG .EQ. 2) ISP3 = 1
ELSE IF (ISPG .EQ. 3) ISP3 = 2
ENDIF
ISP2 = 0
ISP3 = 0
IF (TEHSTF(L,4,J) .EQ. ISP2) ISP2 = 1
IF (TEHSTF(L,5,J) .EQ. ISP2) ISP2 = 2
IF (TEHSTF(L,6,J) .EQ. ISP2) ISP2 = 3
IF (TEHSTF(L,4,J) .EQ. ISP3) ISP3 = 1
IF (TEHSTF(L,5,J) .EQ. ISP3) ISP3 = 2
IF (TEHSTF(L,6,J) .EQ. ISP3) ISP3 = 3

EPSTL1 = EPSX*DBC1,1)*DBC1,1)*EPSY*DBC1,1)*EPSZ*DBC1,1)*
          2  GAHAYX*DBC1,1)*DBC1,1)*EPSY*DBC1,1)*EPSZ*DBC1,1)*
          3  EPSX*DBC1,1)*DBC1,1)*EPSY*DBC1,1)*EPSZ*DBC1,1)*
          1  EPSX*DBC2,1)*DBC2,1)*EPSY*DBC2,1)*EPSZ*DBC2,1)*
          2  EPSX*DBC2,1)*DBC2,1)*EPSY*DBC2,1)*EPSZ*DBC2,1)*
          3  EPSX*DBC2,1)*DBC2,1)*EPSY*DBC2,1)*EPSZ*DBC2,1)*

CHST12 = 0.0
CHST23 = 0.0
CHST13 = 0.0
ELSE IF (TEHSTF(L,ISP,J).EQ.1.0 .AND. TEHSTF(L,ISP2,J).EQ.1.0) THEN
   CHST12 = DSQRT(TEHSTF(L,24+ISP,S)*
             1  HISTO(L,26,J)*TEHSTF(L,24+ISP2,S)*
             2  HISTO(L,26,J))
ELSE IF (TEHSTF(L,ISP,J).EQ.1.0 .AND. TEHSTF(L,ISP2,J).EQ.0.0) THEN
   CHST12 = DSQRT(TEHSTF(L,24+ISP,S)*
             1  HISTO(L,26,J)*HISTO(L,20,J))/
             2  HISTO(L,26,J))
ENDIF
\end{verbatim}
```
CHST12 = 0.0
ELSE IF (TENSTF(L,ISP,J) .EQ. 0.0 .AND. TENSTF(L,ISP2,J) .EQ. 1.0) THEN
  CHST12 = DSQRT(TENSTF(L,24+ISPG,J)/HISTO(L,26,J) + HISTO(L,20,J)/HISTO(L,26,J))
ELSE IF (EPSTL1 .GT. TENSTF(L,9+ISPG,J))
  CHST12 = 0.0
ELSE IF (TENSTF(L,ISP,J) .EQ. 0.0 .AND. TENSTF(L,ISP2,J) .EQ. 1.0) THEN
  CHST12 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J) - HISTO(L,20,J)/HISTO(L,26,J))
ELSE IF (EPSTL2 .GT. TENSTF(L,9+ISPG,J) OR.
  EPSTL1 .GT. TENSTF(L,9+ISPG,J))
  CHST12 = 0.0
ENDIF

CHST13 = 0.0
ELSE IF (TENSTF(L,ISP,J) .EQ. 1.0 .AND. TENSTF(L,ISP3,J) .EQ. 1.0) THEN
  CHST13 = DSQRT(TENSTF(L,24+ISPG,J)/HISTO(L,26,J) - TENSTF(L,24+ISPG3,J)/HISTO(L,26,J))
ELSE IF (EPSTL1 .GT. TENSTF(L,9+ISPG,J))
  CHST13 = 0.0
ELSE IF (TENSTF(L,ISP,J) .EQ. 0.0 .AND.
  TENSTF(L,ISP3,J) .EQ. 0.0) THEN
  CHST13 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J) - HISTO(L,20,J)/HISTO(L,26,J))
ELSE IF (EPSTL3 .GT. TENSTF(L,9+ISPG,J) OR.
  EPSTL1 .GT. TENSTF(L,9+ISPG,J))
  CHST13 = 0.0
ENDIF

CHST23 = 0.0
ELSE IF (TENSTF(L,ISP3,J) .EQ. 1.0 .AND. TENSTF(L,ISP2,J) .EQ. 1.0) THEN
  CHST23 = DSQRT(TENSTF(L,24+ISPG3,J)/HISTO(L,26,J) - TENSTF(L,24+ISPG2,J)/HISTO(L,26,J))
ELSE IF (EPSTL2 .GT. TENSTF(L,9+ISPG,J))
  CHST23 = 0.0
ELSE IF (TENSTF(L,ISP3,J) .EQ. 0.0 .AND. TENSTF(L,ISP2,J) .EQ. 1.0) THEN
  CHST23 = DSQRT(HISTO(L,20,J)/HISTO(L,26,J) - HISTO(L,20,J)/HISTO(L,26,J))
ELSE IF (EPSTL3 .GT. TENSTF(L,9+ISPG,J))
  CHST23 = 0.0
ELSE IF (TENSTF(L,ISP2,J) .EQ. 0.0 .AND.}
```
TESTF(L, ISP3, J) .EQ. 0.0) THEN
CST23 = DSQRHISTO(L, 20, J)/HISTO(L, 26, J)
IF (EPSTL3 .GT. TESTF(L, 9*ISP3, J) .OR. EPSTL2 .GT. TESTF(L, 9*ISP3, J))
CST23 = 0.0
ENDIF

TEMP11(1,1) = CST23*HISTO(L, 26, J)*HISTO(L, 27, J)
TEMP11(1,2) = CST23*HISTO(L, 26, J)*HISTO(L, 27, J)
TEMP11(1,3) = CST23*HISTO(L, 26, J)*HISTO(L, 27, J)
TEMP11(1,4) = CST23*HISTO(L, 26, J)*HISTO(L, 27, J)

CALL PSABS (TESTF, DC, IF)
DO IN = 1, 6
DO IP = 1, 6

C  
END IF
ENDIF
END IF
CONTINUE
END IF

IF (IFLGl .EQ. 1 .AND. IFLG2 .EQ. 1 .AND. IFLG3 .EQ. 1) CALL DCOMP5
(DC0H, IOUT, MA0CRK, MAXCK2, DCRACK, L, J, IELEM)
TEMP11(1,1) = 0.0
TEMP11(1,2) = 0.0
TEMP11(1,3) = 0.0
DO 180 IK = 1, 6
DO 170 IP = 1, 6

DC0H(IP, IK) = DC0N(IP, IK) + DSTIF(IP, IK) + TEMP11(IK, IP)

CONTINUE

STORE THE GLOBAL CONSTITUTIVE MATRIX RETURNED IN 'DC0H'.
CRACK(L,20,J) = DCDH(4,1)
CRACK(L,21,J) = DCDH(4,2)
CRACK(L,22,J) = DCDH(4,3)
CRACK(L,23,J) = DCDH(4,4)
CRACK(L,24,J) = DCDH(4,5)

CRACK(L,25,J) = DCDH(4,6)
CRACK(L,26,J) = DCDH(4,7)
CRACK(L,27,J) = DCDH(4,8)
CRACK(L,28,J) = DCDH(4,9)
CRACK(L,29,J) = DCDH(4,10)
CRACK(L,30,J) = DCDH(4,11)

CRACK(L,31,J) = DCDH(4,12)
CRACK(L,32,J) = DCDH(4,13)
CRACK(L,33,J) = DCDH(4,14)
CRACK(L,34,J) = DCDH(4,15)
CRACK(L,35,J) = DCDH(4,16)
CRACK(L,36,J) = DCDH(4,17)

CRACK(L,37,J) = DCDH(4,18)
CRACK(L,38,J) = DCDH(4,19)
CRACK(L,39,J) = DCDH(4,20)
CRACK(L,40,J) = DCDH(4,21)
CRACK(L,41,J) = DCDH(4,22)

CRACK(L,42,J) = DCDH(4,23)
CRACK(L,43,J) = DCDH(4,24)

RETURN
C
C

C ------------------------------- D C O M P S -------------------------------
C
C INCLUDE(PROCESS)
SUBROUTINE DCOMPS(DCOB, JOUT, MAXCRK, MAXCK2, DCRACK, L, JELM)
IMPLICIT REAL*(A-H,0-Z)
C...SWITCHES: RESUMB=100:10,FORMAT=900:10
C...SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C SIMU
C
C*******************************************************************
COMMON /BLOCKS/HISTO(27,70,15),CRACK(27,250,15),HISTY,IBIST
$ ,IBIST2
DIMENSION DCOB(6,*) ,DCRACK(MAXCK2,*) ,DB(6,30),DDBCK2(6,30),
$ DSCMBT(6,6)
REAL*(H(6,30),MTRABS(30,6),MTD(30,30),ETDN(30,30)
EQUIVALENT (H,DDBCK2)
BCRACK = INT(CRACK(1,1))
C
C INITIALIZE THE NECESSARY ARRAYS.
C
C DO 120 II = 1, 3*BCRACK
DO 100 JJ = 1, 6
MTD(II,JI) = 0.0
MTRABS(II,JI) = 0.0
$ (JJ,II) = 0.0
DE(JJ,II) = 0.0
100 CONTINUE
DO 110 KK = 1, 3*BCRACK
DCRACK(II,KK) = 0.0
MTDN(II,KK) = 0.0
110 CONTINUE
120 CONTINUE
C
C GENERATE TRANSFORMATION MATRIX 'H(3,2*(NO. OF CRACKS))'
DO 130 I = 1, NCRACK
K = 3*(I-1) + 1

THETA = ORIENTATION OF EACH CRACK (COUNTER-CLOCKWISE FROM THE GLOBAL X TO THE NORMAL OF THE CRACK).

ICK = 9*(I-1) + 1
AL1 = CRACK(I,137+ICK,J)
AM1 = CRACK(I,138+ICK,J)
AM2 = CRACK(I,141+ICK,J)
AH2 = CRACK(I,142+ICK,J)
AL3 = CRACK(I,143+ICK,J)
AH3 = CRACK(I,146+ICK,J)

AM1 = CRACK(L,137+ICK,J)
AH1 = CRACK(L,138+ICK,J)
AH2 = CRACK(L,141+ICK,J)
AL3 = CRACK(L,143+ICK,J)
AH3 = CRACK(L,146+ICK,J)

H(1,K) = AL1*AL1
H(1,K+1) = AL1+AL2
H(1,K+2) = AL1+AL3

H(2,K) = AM1*AM1
H(2,K+1) = AM1*AM2
H(2,K+2) = AM1*AM3

H(3,K) = AH1*AH1
H(3,K+1) = AH1*AH2
H(3,K+2) = AH1*AH3

H(4,K) = 2.0*AL1*AM1
H(4,K+1) = AL1*AM2 + AL2*AH1
H(4,K+2) = AL1*AM3 + AL2*AH2

H(5,K) = 2.0*AM1*AH1
H(5,K+1) = AM1*AH2 + AM2*AH1
H(5,K+2) = AM1*AH3 + AM2*AH2

H(6,K) = 2.0*AH1*AL1
H(6,K+1) = AH1*AL2 + AH2*AL1
H(6,K+2) = AH1*AL3 + AH2*AL2

GENERATE TRANSPOSE OF 'H'.

MTRANS(K,1) = H(1,K)
MTRANS(K+1,1) = H(1,K+1)
MTRANS(K+2,1) H(1,K+2)
MTRANS(K,2) = H(2,K)
MTRANS(K+1,2) = H(2,K+1)
MTRANS(K+2,2) = H(2,K+2)
MTRANS(K,3) = H(3,K)
MTRANS(K+1,3) = H(3,K+1)
MTRANS(K+2,3) = H(3,K+2)
MTRANS(K,4) = H(4,K)
MTRANS(K+1,4) = H(4,K+1)
MTRANS(K+2,4) = H(4,K+2)
MTRANS(K,5) = H(5,K)
MTRANS(K+1,5) = H(5,K+1)
MTRANS(K+2,5) = H(5,K+2)
MTRANS(K,6) = H(6,K)
MTRANS(K+1,6) = H(6,K+1)
MTRANS(K+2,6) = H(6,K+2)

DETERMINE D(CRACKS) MATRIX CONTAINING DIRECT AND SHEAR STIFFNESSES OF EACH CRACK ON ITS DIAGONAL.

SCRACK = STIFFNESS ACROSS EACH CRACK.
GCRACK = ; ALONG ; ; .
SCRACK = CRACK(L,IS+I,J)
GCRCK1 = CRACK(L,97+I,J)
GCRCK2 = CRACK(L,107+1,J)
DCRACK(K,K) = SCRACK
DCRACK(K+1,K+1) = GCRCK1
DCRACK(K+2,K+2) = GCRCK2

130 CONTINUE

C
C PERFORM HTRANS = H(TRANSPOSE) * D(CON.) MULTIPLICATION
C
D  160 1=1, 3+HCRACK
DO 150 K = 1, 6
DO 140 LL = 1, 6
HTRANS(I,K) = HTRANS(I,K) + HTRANS(I,LL) * DCON(LL,K)
140 CONTINUE
150 CONTINUE
160 CONTINUE

C
C PERFORM D N = D(CON.) * N MULTIPLICATION.
C
D  180 1 = 1, 3*NCRACK
DO 170 K = 1, 3*NCRACK
DO 160 LL = 1, 6
D(N)(1,K) = D(N)(I,K) + DCDN(I,Ll) * H(LL,K)
160 CONTINUE
170 CONTINUE
180 CONTINUE
190 CONTINUE

C
C PERFORM DTRANS = HTRANS * D(CON.) * H MULTIPLICATION
C
D  210 1 = 1, 3+HCRACK
DO 200 K = 1, 3+HCRACK
DO 190 LL = 1, 6
HTRANS(I,K) = HTRANS(I,K) + HTRANS(I,LL) * H(LL,K)
190 CONTINUE
200 CONTINUE
210 CONTINUE
220 CONTINUE
DO 240 I = 1, 3+HCRACK
DO 230 K = 1, 3+HCRACK
DTRANS(I,K) = DTRANS(I,K) + HTRANS(I,K)
230 CONTINUE
240 CONTINUE

240 CONTINUE

240 CONTINUE

C
C INVERT THE 'DCRACK' MATRIX.
C
H = NO. OF ROWS AND COLUMNS IS 'DCRACK'.

C
C NROCOL = 3*NCRACK
C
C NRC = MAX. NO. OF ROWS (AND COLUMNS) ALLOWED.
C
C EPS = MIN. ALLOWABLE MAGNITUDE FOR A PIVOT ELEMENT.
C
C EPS = 10.0E-20
C
C INVERSE OF 'DCRACK' WILL BE RETURNED IN 'DCRACK'.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
DETER IS THE DETERMINANT OF 'DCRACK' THAT IS RETURNED
AS THE VALUE OF FUNCTION 'SIMUL'.

DETER = SIMUL(BRICKL, DCRACK, EPS, BRC, IOUT)
IF (DETER .EQ. 0 . ) THEN
  WRITE (IOUT, 900) L, J, IELEM
  STOP
ENDIF
DO 260 I = 1, 6
  DO 250 K = 1, 3*BCRACK
    DBDCRI(I,K) = 0.0
  CONTINUE
250 CONTINUE
260 CONTINUE

PERFORM

DBDCRI = DS * INVERSE OF DCRACK MULTIPLICATION

DO 290 I = 1, 6
  DO 280 K = 1, 3*BCRACK
    DO 270 LL = 1, 3*BCRACK
      DBDCRI(I,K) = DBDCRI(I,K) + DBDCL(I,LL)*DCRACK(LL,K)
    CONTINUE
270 CONTINUE
280 CONTINUE
290 CONTINUE

INITIALIZE MATRIX 'DSUBCT'. THIS MATRIX MUST BE SUBTRACTED
FROM THE INTACT CONCRETE CONSTITUTIVE MATRIX TO RESULT I
THE CRACKED-CONCRETE CONSTITUTIVE LAW.

DO 310 I = 1, 6
  DO 300 K = 1, 6
    DSUBCT(I,K) = 0.0
  CONTINUE
300 CONTINUE
310 CONTINUE

COMPUTE

DSUBCT = DBDCRI * NTD

DO 340 I = 1, 6
  DO 330 K = 1, 6
    DO 320 LL = 1, 3*NCRACK
      DSUBCT(I,K) = DSUBCT(I,K) + DBDCRI(I,LL)*NDT(LL,K)
    CONTINUE
320 CONTINUE
330 CONTINUE
340 CONTINUE

GENERATE THE CONSTITUTIVE MATRIX FOR CRACKED CONCRETE.

DO 360 I = 1, 6
  DO 350 K = 1, 6
    DCON(I,K) = DCON(I,K) - DSUBCT(I,K)
  CONTINUE
350 CONTINUE
360 CONTINUE
RETURN
900 FORMAT(/'ERROR'/, 6I2, 'DURING CONSTITUTIVE MATRIX', 
  1 'INVERSION A NEGATIVE PIVOT WAS ENCOUNTERED', 6I2, 
  2 'LAYER', I2, 'IELEM', I3)
END
SUBROUTINE STATUS(STRESS, STATE, J, PROPER, IFC)
IMPLICIT REAL*8(A-H,Q-Z)

C . .SWITCHES:  REBUHB=100:10, FDRMAT=900:10
C , . .SWITCHES:
C**************************************************************
C COMMON /BLOCKS/HISTO(27,70,15),CRACK(27,280,16),IHISTY,IHIST1
$ ,IHIST2
DIMENSION PROPER(25)
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C**************************************************************

SUBROUTINE TO DETERMINE THE STATE OF BIAXIAL STRESS.

..... DEFINE INFINITY 'CINFIT'

C
CINFIT = 10.0E+30
FC = PROPER(6)
FT = PROPER(6)
IF (STRESS .GE. STRESS) THEN
  IF (STRESS .GE. STRESS) THEN
    SIGMA1 = STRESS
    IF (STRESS .GE. STRESS) THEN
      SIGMA2 = STRESS
      SIGMA3 = STRESS
    ELSE
      SIGMA2 = STRESS
      SIGMA3 = STRESS
    ENDIF
  ELSE
    SIGMA1 = STRESS
    SIGMA2 = STRESS
    SIGMA3 = STRESS
  ENDIF
ELSE IF (STRESS .GE. STRESS) THEN
  SIGMA1 = STRESS
  SIGMA2 = STRESS
  SIGMA3 = STRESS
ELSE
  SIGMA1 = STRESS
  SIGMA2 = STRESS
  SIGMA3 = STRESS
ENDIF
C
C IF (FC.EQ.2 .AND. CRACK(L,1,J).GE.1.0) THEN
  IF (HISTO(L,41,J).LT.PROPER(7)) THEN
    FC = FC*(1.0-0.1*(HISTO(L,41,J)/PROPER(7)))
  ELSE
    FC = FC*0.8
  ENDIF
ENDIF
$$\text{FC} = \text{DMIE}(_{\text{HISTO}}(L,58,J),\text{FC})$$

$\text{HISTO}(L,58,J) = \text{FC}$

C STATE OF UNIAXIAL TENSION.
C
IF ($\text{SIGMA}_1 \geq -0.0001$ AND $\text{SIGMA}_2 \leq 0.0001$ AND $\text{SIGMA}_3 \geq -1.0001$ AND $\text{SIGMA}_3 \leq 0.0001$ AND $\text{SIGMA}_1 \geq 0.0001$) STATE = 100.0
C STATE OF BIAXIAL TENSION.
C
IF ($\text{SIGMA}_1 \gt 0.0$ AND $\text{SIGMA}_2 \gt 0.0$ AND $\text{SIGMA}_3 \leq -0.73333 \times \text{RC}$) THEN
  R = $\text{SIGMA}_1 / \text{SIGMA}_3$
  RC = $\text{FT} / \text{FC}$
  IF (R $\gt -0.7333 \times \text{RC}$) THEN
    STATE = 101.0
  ELSE IF (R $\leq -0.7333 \times \text{RC}$) THEN
    STATE = -101.0
  ENDIF
ENDIF
C STATE OF TRIAXIAL TENSION.
C
IF ($\text{SIGMA}_1 \lt 0.0$ AND $\text{SIGMA}_2 \leq 0.0001$ AND $\text{SIGMA}_3 \geq -1.0001$ AND $\text{SIGMA}_3 \leq 0.0001$ AND $\text{SIGMA}_1 \gt 0.0001$) STATE = 111.0
C STATE OF UNIAXIAL COMPRESSION.
C
IF ($\text{SIGMA}_1 \lt 0.0$ AND $\text{SIGMA}_2 \geq -0.0001$ AND $\text{SIGMA}_2 \leq 0.0001$ AND $\text{SIGMA}_1 \geq 0.0001$ AND $\text{SIGMA}_1 \leq -1.0001$) STATE = -100.0
C STATE OF BIAXIAL COMPRESSION.
C
IF ($\text{SIGMA}_2 \lt 0.0$ AND $\text{SIGMA}_3 \lt 0.0$ AND $\text{SIGMA}_1 \geq 0.0001$ AND $\text{SIGMA}_1 \leq -1.0001$) STATE = -110.0
C STATE OF TRIAXIAL COMPRESSION.
C
IF ($\text{SIGMA}_1 \lt 0.0$ AND $\text{SIGMA}_2 \lt 0.0$ AND $\text{SIGMA}_3 \lt 0.0$) STATE = -111.0
RETURN
END
C
C ----------------------------------------------- MATERIAL -------------------------------------------------
C
INCLUDE(PROCESS)
SUBROUTINE MATERIAL(EGUS,MLAYER,SIGMA1,SIGMA2,SIGMA3,STATE, 
$\text{ES},\text{MUS},\text{IOUT},\text{PROPER},\text{BETHA},\text{IFC},\text{TELEM})$
IMPLICIT REAL*(-1,-0,-2)
C...SWITCHES: NENUMB=100:10,FORMAT=900:10
C...SWITCHES:
SUBROUTINE TO DETERMINE SECANT VALUES OF E AND NUG
FOR THE CURRENT STATE OF STRESS, BEFORE ULTIMATE &
AT THE ULTIMATE.

IFG = 0.0
AC = 0.0
BC = 0.0
FR1 = 0.0
FR2 = 0.0
SIGIN = 0.0
SIG2N = 0.0
SIG3N = 0.0
FC = PROPER(S)
FT = PROPER(6)
IF (CRACK(NGUS, 1, NLAYER) .GE. 1.0) THEN
   IF (IFC .EQ. 2) THEN
      IF (HISTOCHGUS,41,NLAYER) .LT. PROPER(7)) THEN
         FC = FC*(1.0-O.2*CHIST0(NGUS,41,NLAYER)/PROPER(7)))
      ELSE
         FC = FC*0.5
      ENDIF
   EHDIF
   FC = DMINKHISTO(NGUS,58,NLAYER),FC)
   FC = HISTOCHGUS,58,NLAYER)
   IF (STATE.EQ.100. .OR. STATE. EQ.110. .OR. STATE.EQ.111.) THEN
      C STATE OF UNIAXIAL, BIAXIAL OR TRIAXIAL TENSION.
      C BETHA = EOSLINEARITY INDEX.
      IFG = 1
      AAA = -1000.0*FC
      BBB = 0.0
      CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGM3F,
      IDOUT, STATE, SIGMIF, SIGM2F)
      C CALCULATE SECANT VALUES OF E AND NUG.
      IF (BETHA .GT. 1.0) THEN
         WRITE (*, *) 'SIGMA1', SIGMA1, 'SIGMA2', SIGMA2, 'SIGMA3'
         , SIGMA3
         WRITE (*, *) 'SIGMA1F', SIGM1F, 'SIGMA2F', SIGM2F, 'SIGMA3F'
         1, SIGM3F
         WRITE (*, *) '1 = LAYER', NLAYER, 'GPRT', NGUS, 'ELEM',
         1, IELEM
      EHDIF
CALL TEBSIL (ES, BUS, BETHA, IOUT, PROPER, BLAYER, BGUS, IFC, SIGMA1, SIGMA2, SIGM3F, IELEM)

ENDIF

IF (STATE .EQ. 101) THEN

STATE OF TENSION-COMPRESSION (CRACKING ZONE)

TOLERANCE = .0001

IFG = 1
AAA = -1000.0*FC
BBB = 0.0
CALL ENVELPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGM3F, IOUT, STATE, SIGMIF, SIGH2F)

C
C CALCULATE SECANT VALUES OF 'E' AND 'E'.
C
IF (BETHA .GT. 1.0) THEN
WRITE (*, *) 'SIGMA1', SIGMA1, 'SIGMA2', SIGMA2, 'SIGMA3', SIGMA3
WRITE (*, *) 'SIGMA4F', SIGMIF, 'SIGMA5F', SIGH2F
WRITE (*, *) '2 # LAYER', BLAYER, 'GPRT', BGUS, 'ELEM', IELEM
ENDIF
CALL TEBSIL (ES, BUS, BETHA, IOUT, PROPER, BLAYER, BGUS, IFC, SIGMA1, SIGMA2, SIGM3F, IELEM)

C
C STATE OF TENSION-COMPRESSION (CRUSHING ZONE)
C
IFG = 1
AAA = -1000.0*FC
BBB = 0.0
CALL ENVELPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGM3F, IOUT, STATE, SIGMIF, SIGH2F)

C
C CALCULATE SECANT VALUES OF 'E' AND 'E'.
C
IF (BETHA .GT. 1.0) THEN
WRITE (*, *) 'SIGMA1', SIGMA1, 'SIGMA2', SIGMA2, 'SIGMA3', SIGMA3
WRITE (*, *) 'SIGMA4F', SIGMIF, 'SIGMA5F', SIGH2F
WRITE (*, *) '3 # LAYER', BLAYER, 'GPRT', BGUS, 'ELEM', IELEM
ENDIF
CALL TEBSIL (ES, BUS, BETHA, IOUT, PROPER, BLAYER, BGUS, IFC, SIGMA1, SIGMA2, SIGM3F, IELEM)

ENDIF

IF (STATE.EQ.(-101.:)) THEN

C
C STATE OF UNIAXIAL/BIAXIAL/TRIAXIAL COMPRESSION.
C
C CALCULATE THE FAILURE VALUE OF SIGMA2 WHEN SIGMA1 IS
C
KEEP CONSTANT.
C
IFG = 1
AAA = -1000.0*FC
BBB = 0.0
CALL ENVELPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIGM3F,
RETRIEVE OTHER LAYER PROPERTIES

CALL LAYPRP (PROPER, HLAYER, FC, FT, BETHAA, EPSC, EI, HUI,

IU, BUF, DPOST, Csigma)

IF (CRACK(BGUS,1,HLAYER) .GE. 1.0) THEN
  IF (IFC .EQ. 2) THEN
    IF (HISTO(BGUS,41,HLAYER) .LT. PROPER(7)) THEN
      FC = FC* (1.0 - 0.2*(HISTO(BGUS,41,HLAYER)/PROPER(7)))
      EPSC = EPSC*(1.0 + 0.10*(HISTO(BGUS,41,HLAYER)/
                          PROPER(7)))
    ELSE
      FC = FC*0.8
      EPSC = EPSC*1.10
    ENDIF
  ENDIF
  FC = DMINI(HISTO(BGUS,68,HLAYER), FC)
  FC = HISTO(BGUS,68,HLAYER)
ENDIF

EC = FC/EPSC
A = EI/EC

CALCULATE THE SECOND IN Variant OF THE DEVIATORIC STRESS
CORRESPONDING TO THE STRESS STATE 'SIGMA1' 'SIGMA2' & 'SIGMA3'.

J2 = (SIGMA1+SIGMA1+SIGMA2+SIGMA2+SIGMA3+SIGMA3-SIGMA1*SIGMA2-
     SIGMA2*SIGMA3*SIGMA3-SIGMA1*SIGMA2-SIGMA1*SIGMA3)/3.0
Z = DSQRT(J2)/FC - 1./DSQRT(3.000)

EF = SECANT VALUE OF YOUNG'S MODULUS AT GENERAL
STATE OF BIAXIAL COMPRESSIVE FAILURE.

EF = EC/(1.+4.*(A-1.)*X)

CALCULATE SECANT VALUES OF 'E' & 'BUSH'.

ES1 = EI/2. - BETHA*(EI/2.-EF)
ES2 = DSQRT(DABS(ES1+ES1+ES1+ES1+EF)*BETHA*(DPOST*(1.-BETHA)-1.))
IF (BETHA .GT. 1.0) THEN
  WRITE (*, *) 'SIGMA1', SIGMA1, 'SIGMA2', SIGMA2, 'SIGMA3'
  , SIGMA3
  WRITE (*, *) 'SIGMA1F', SIGMA1, 'SIGMA2F', SIGMA2, 'SIGMA3F'
  , SIGMA3F
  WRITE (*, *) 'ES # LAYER', HLAYER, 'GPST', BGUS, 'ELEM',
  ILELEM
ENDIF

CALL BFAILR (BETHA, IOUT, ES1, ES2, BETHAA, HUI, BUF, ES, HUS

IU, BGUS, HLAYER, ILELEM)

ENDIF
RETURN
END
C INCLUDE(PROCESS)
SUBROUTINE LAYPRP(ES,BUS,BETA,IOUT,PROPER,BLAYER,BGUS,IFC,SIGM1F,
$SIGM2F,SIGM3F,LELEM)
C IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: REBUMB=100:10,FORMAT=900:10
C...SWITCHES:
C******************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C LAYPRP  BFAILR
C******************************************************************************

C COMMON/BLOCKS/HISTO(27,70,16),CRACK(27,160,16),IHISTY,IHIST1
$ ,IHIST2
DIMENSION PROPER(28)
REAL*8 NUX,NBF,NUS,J2
CALL LAYPRP (PROPER, BLAYER, FC, FT, BETHAA, EPSC, EI, BUI, BUF,
1 DPOST, CSIGMA)
C
EF=SECANT VALUE OF YOUNG'S MODULUS AT UNIAXIAL
COMPRESSIVE FAILURE.
C
IF (CRACK(BGUS,1,BLAYER) .GE. 1.0) THEN
IF (IFC .EQ. 2) THEN
IF (HISTO(BGUS,41,BLAYER) .LT. PROPER(7)) THEN
FC = FC*(1.0-0.2*(HISTO(BGUS,41,BLAYER)/PROPER(7))
EPSC = EPSC*(1.0+0.10*(HISTO(BGUS,41,BLAYER)/PROPER(7)
1)
ELSE
FC = FC*0.8
EPSC = EPSC*1.10
ENDIF
ENDIF
ENDIF
C EF = SECANT VALUE OF YOUNG'S MODULUS AT GENERAL
STATE OF BIAXIAL COMPRESSIVE FAILURE.
C
EC = FC/EPSC
EF = EC
C CALCULATE SECANT VALUES OF 'E' & 'NU'.
C
ES1 = EI/2. - BETA*(EI/2.-EF)
ES2 = DSQRT(ES1*ES1*(EF*EF)+BETA*(DPOST*(1.-BETA)-1.))
CALL BFAILR (BETHA, IOUT, ES1, ES2, BETHAA, HUI, BUF, ES, BUS,
1 BGUS, BLAYER, IELEM)
RETURN
END
C
C L A Y P R P
C
C INCLUDE(PROCESS)
SUBROUTINE LAYPRP(ES,BUS,BETA,IOUT,PROPER,BLAYER,BGUS,IFC,SIGM1F,
$SIGM2F,SIGM3F,LELEM)
SUBROUTINE TO PROVIDE MATERIAL PROPERTIES FOR THE CURRENT CONCRETE LAYER.

FC = PROPER(6)
FT = PROPER(6)
BETHAA = PROPER(13)
EPSC = PROPER(7)
EI = PROPER(1)
NUI = PROPER(2)
BUF = PROPER(12)
DPST = PROPER(9)
CEIGMA = PROPER(14)
RETURN
END

SUBROUTINE TO CALCULATE CURRENT SECANT E & POISSON'S RATIO WHEN THE STATE OF STRESS IS INSIDE THE FAILURE

IF (BETHA .GE. 1.01) THEN
WRITE (1OUT, 900) BETHA, NGUS, LYRHO, IELEM
STOP
ENDIF
EM = ES1 - ES2
ES = ES1 + ES2
HISTO(NGUS, ES, LYRHO) = EM
IF (BETHA .LE. BETHAA) HUS = HUI
IF (BETHA .GT. BETHAA) BUS = HUF - (HUF-HUI)*DSQRT(DABS(1.0-(BETHA-BETHAA)/(1.0-BETHAA)) **2.0))
RETURN
900 FORMAT(/,10X,'THE NONLINEARITY INDEX IS GREATER THAN 1.0 ',
1 'WHEN THE STRESS STATE IS INSIDE THE ',
2 'ENVELOPE 3',/30X,'BETHA=',DI2.5,'FOR G.PT',I2,'LAYER'
3 ,I2,'ELEMENT',I3)
END
C
C ------------------------------- FAILUR -------------------------------
C INCLUDE(PROCESS)
C SUBROUTINE FAILUR(L,LXRO,PROPER,IGP,ULSUS,USUSL,SIGMA1,SIGMA2
1 $ ,SIGMA3,IFC,ELEH)
IMPLICIT REAL*8(A-H,O-Z)
C  .SWITCHES: REHUMB=100:10,F0RHAT=900:10
C  .SWITCHES:
C SUBROUTINES AND FUNCTION CALLS FROM THIS ROUTINE
C EHLPE MATERL
C
C******************************************************************************
C COMMON /BLOCK5/HIST0(27,70,18),CRACK(27,260,18),HIST0,HE1ST
$ ,HE1ST
C COMMON /ABC/TEHSTF(27,80,16),ITHSG0,ITHSG1,ITHSG2
C COMMON /OTTO/ AC,BC,FK1,FK2,SIG1H,SIG2H,SIG3H,FC,AAA,BBB,FT,IFG
DIMEHSIOH PROPER(28)
REAL*8 USUSL
C
C FOR EACH GAUSS POINT DETERMINE THE FAILURE VALUES OF 'SIGMA1',
C OF 'SIGMA2', 'SIGMA3' GIVEN THE STATE OF STRESS
C AND THE RATIO OF STRESSES.THIS RATIO IS THE CURRENT
C ONE FOR THE STRESS POINT THAT HAS VIOLATED THE
C FAILURE ENVELOPE OR MAY BE ON IT.
C
C IFG = 0.0
AC = 0.0
BC = 0.0
FK1 = 0.0
FK2 = 0.0
SIG1H = 0.0
SIG2H = 0.0
SIG3H = 0.0
STATE = HIST0(L,26,LYRHO)
FC = PROPER(8)
FT = PROPER(6)
IF (CRACK(L,1,LYRHO) .GE. 1.0) THEN
   IF (IFC .EQ. 2) THEN
      IF (HIST0(L,41,LYRHO) .LT. PROPER(7)) THEN
         FC = FC*(1.0-0.2*(HIST0(L,41,LYRHO)/PROPER(7)))
      ELSE
         FC = FC*0.8
      ENDIF
   ELSE
      FC = DMIN1(HIST0(L,56,LYRHO),FC)
   ENDDIF
ENDIF
ENDIF
FC = DMIN1(HIST0(L,56,LYRHO),FC)
HISTO(L,SB,LYRD) = FC
IF (STATE .EQ. 100.) THEN
   C STATE OF UNIAXIAL TENSION.
   SIGM1F = FT
   SIGM2F = 0.0
   SIGM3F = 0.0
   GO TO 100
ENDIF
IF (STATE .EQ. 110.) THEN
   C STATE OF BIAXIAL TENSION.
   SIGM1F = FT
   RATIO = SIGMA2/SIGMA1
   SIGM2F = SIGM1F*RATIO
   SIGM3F = 0.0
   GO TO 100
ENDIF
IF (STATE .EQ. 111.) THEN
   C STATE OF TRIAXIAL TENSION.
   SIGM1F = FT
   RATIO1 = SIGMA2/SIGMA1
   RATIO2 = SIGMA3/SIGMA1
   SIGM2F = SIGM1F*RATIO1
   SIGM3F = SIGM1F*RATIO2
   GO TO 100
ENDIF
IF (STATE .EQ. 101) THEN
   C STATE OF TENSION-COMPRESSION (CRACKING ZONE)
   IFG = 2
   AAA = 0.0
   BBB = FT*1.5
   C.. NOTE STRESSES SUPPLIED INVERSELY TO RECORD THE TENSILE CRACKING
   C STRESS LESS THAN OR EQUAL TO FT.
   CALL ENVLPF (SIGMA, SIGMA2, SIGMA3, PROPER, BETA, SIG3F, 1
               IOUT, STATE, SIGM1F, SIGM2F)
   SIGM1F = SIGIF
   SIGM2F = SIGIF+SIGM2F/SIGMA1
   SIGM3F = SIGM3F+SIGM3F/SIGMA3
   GO TO 100
ENDIF
IF (STATE.EQ.(-101.)) THEN
   C STATE OF TRAVERSE-COMPRESS (CRUSHING ZONE)
   IFG = 2
   AAA = -1000.0*FC
   BBB = 0.0
   CALL ENVLPF (SIGMA, SIGMA2, SIGMA3, PROPER, BETA, SIG3F, 1
               IOUT, STATE, SIGM1F, SIGM2F)
   SIGM3F = SIG3F
   SIGM2F = SIGM3F+SIGM2F/SIGMA2
   SIGM1F = SIG3F+SIGMA3/SIGMA1
   ENDIF
IF (STATE.EQ.(-100.) .OR. STATE.EQ.(-110.) .OR. STATE.EQ.(-111.) THEN
   C STATE OF UNIAXIAL OR BI/TRI AXIAL COMPRESSION.
AAA = -1000.0*FC
BBB = 0.0
FFG = 2
CALL ENVLPE (SIGMA1, SIGMA2, SIGMA3, PROPER, BETHA, SIG3F,
1
IOUT, STATE, SIGMIF, SIGM2F)
SIGM3F = SIG3F
SIGM3F = SIGM3F*(SIGMA2/SIGMA3)
SIGM3F = SIG3F*(SIGMA1/SIGMA3)
ENDIF

DETERMINE MATERIAL PARAMETERS AT FAILURE.

DETERMINE BULK MODULUS AT FAILURE.

SAVE PRINCIPAL STRESSES AT ULTIMATE.

RETURN
END
IF (IFG .EQ. 1) THEN
  IF (SIGMA1 .GE. 0.0) THEN
    SIGIH = 0.0
    SIG2H = SIGMA2 - SIGMA1
    SIG3H = SIGMA3 - SIGMA1
  ELSE
    SIGIH = SIGMA1
    SIG2H = SIGMA2
    SIG3H = SIGMA3
  ENDIF
ELSE IF (IFG .EQ. 2) THEN
  SIGIH = SIGMA1
  SIG2H = SIGMA2
  SIG3H = SIGMA3
ENDIF

C FIND THE ROOTS
CALL ROOT (FVAL, A, B, HQ, T2, E, BOUNDS, HROOT, WS, WS1, L)

C FIND THE FAILURE VALUE OF STRESS FOR GIVEN SIGMA 1 AND 2
IF (HROOT .GT. 10) WRITE (TOUT, *) 'ROOTS EXCEEDED MAX ENVELOPE'
IF (HROOT .EQ. 0) WRITE (TOUT, *) 'NO ROOTS IN BOUND A', A, 'B'
DO 100 IM = 1, HROOT
  XI = 2*IM - 1
  X2 = 3*IM
  ALDB(IM) = BOUNDS(X1)
  AUDB(IM) = BOUNDS(X2)
  IPERM1(IM) = IM
  IPERM2(IM) = IM
100 CONTINUE
IF (HROOT .GT. 0) THEN
  CALL DSVRGP (HROOT, ALDB, ALDB, IPERM1)
  CALL DSVRGP (HROOT, AUDB, AUDB, IPERM2)
ENDIF
DO 110 IM = 1, HROOT
  IF (IPERM1(IM) .NE. IPERM2(IM)) WRITE (HROOT, *)
     'PERMUTATION MISTAKE IN ENVELOPE'
    ALDB(IM) = (ALDB(IM)+AUDB(IM))/2.0
110 CONTINUE
IF (STATE.EQ.101 .AND. IFG.EQ.2) GO TO 140
DO 120 IPF = HROOT, 1, -1
  IF (DABS(SIG3H) .LE. DABS(ALDB(IPF))) THEN
    SIG3F = ALDB(IPF)
  GO TO 130
ENDIF

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ENDIF

120 CONTINUE
IF (SIGM3F .EQ. 0.0) THEN
  WRITE (*,*) 'FAILURE VALUE NOT FOUND IFG', IFG
  WRITE (*,*) 'SIG1H', SIG1H, 'SIG2H', SIG2H, 'SIG3H', SIG3H
  SIGM3F = -1.18*FCP
  WRITE (*,*) 'FC', FCP, 'SIG3H', SIG3H
ENDIF

C

130 CONTINUE
IF (IFG .EQ. 1) BETHA = DABS(SIG3H)/DABS(SIGM3F)
IF (BETHA .GT. 1.0) WRITE (*, 900)
C

140 CONTINUE
IF (STATE.EQ.101 .AND. IFG.EQ.2) THEN
  DO ISO IHH = 1, HROOT
    IF (ALDB(IHH).GT.0.0 .AND. ALDB(IHH).LE.FT) THEN
      SIGMIF = ALDB(IHH)
    ENDIF
  ENDDO
RETURN
ENDIF

150 CONTINUE
IF (SIGMIF .EQ. 0.0) THEN
  WRITE (*,*) 'FAILURE VALUE OBTAINED IFG', IFG
  WRITE (*,*) 'SIG1H', SIG1H, 'SIG2H', SIG2H, 'SIG3H', SIG3H
  WRITE (*,*) 'SETTING SIGM3F TO BE SIG3H-TENSION-CUTOFF'
  WRITE (*,*) 'FT', FT, 'FVAL', ALDB(HROOT)
  SIGMIF = FT
ENDIF
ENDIF
RETURN

900 FORMAT(1X,'LOAD INCREMENT SIZE TOO LARGE REFINE STEP SIZE',
       1 'CHECK VALUES OF GF AND TENSION STIFFENING LAYER ALSO',
       2 'THESE QUANTITIES MAY BE SMALL LEADING TO FALSE CRUSHING')
END

C ***********************************************
C INCLUDE(PROCESS)
C SUBROUTINE GOSTEN(FCP,FCT,PROPER,OUT)
IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: REDUCE=100:10,FORMAT=900:10
C...SWITCHES:
C ***********************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C ***********************************************
DIMENSION PROPER(25)
AK = FCT/FCP
C X AND Y ARE DEFINED BASED ON BALMER(1929) TRIAXIAL TEST DATA
C (X,Y) =(-6/DQSQRT3,4/DQSQRT3) FOR SCHICKERT DATA USE 3.28 FOR 4
C X = -50.0/FCP
C Y = 43.61/FCP
C X = 8.0 / DQSQRT(3.0D00)
C Y = 3.28 / DQSQRT(3.0D00)
C
DI-AXIAL STRENGTH DATA 1.16 FOR KUPFER DATA AND 1.21 FOR SCHICHERT DATA.

\[
SIGBC = 1.16
\]

\[
SIGBC = 1.21
\]

\[
H = -\frac{DSQRT(2.0DO0)*X*Y)}{(Y/DSQRT(2.0DO0)-1./3.)}
B = \frac{(DSQRT(2.0DO0)-(3.*Y)/(AK+SIGBC))/(H-(0.0*Y)/(SIGBC-AK))}{Y}
A = \frac{(B*B-DSQRT(2.0DO0))}{Y}
\]

\[
ALAMBC = (1.0-H/(3.*Y))\times DSQRT(3.0DO0)+ B + DSQRT(3.0DO0) + DSQRT(1.0DO0)/(DSQRT(3.0DO0)+Y)
\]

\[
ALAMBT = (2.*DSQRT(3.0DO0)-(SIGBC*H)/(DSQRT(3.0DO0)*Y))\times Y + (DSQRT(2.0DO0)*SIGBC)/(DSQRT(3.0DO0)*Y) + DSQRT(3.0DO0)/SIGBC
\]

\[
FK2 = DCOS(3.0*DATAD(2.0*ALAMBC/ALAMBT-1.0)/DSQRT(3.0DO0))
FK1 = ALAMBT/DCOS(1.0DO0/3.0DO0)*DCOS(FK2)
\]

\[
PRDPERC18) = A
PR0PERC19) = B
PRDPERC20) = FK1
PROPER(21) = FK2
\]

RETURN

END

***********************************************************************
ROOT ISOLATION PROGRAM
***********************************************************************

PROGRAM TO ISOLATE THE ROOTS FOR A TRANSCENDENTAL EQUATION

***********************************************************************

INCLUDE(PROCESS)

SUBROUTINE ROOICFVAL,A, B,KQ,T2.E,BOUNDS, NROOT, HS, WSL,L)

IMPLICIT REAL*8(A-H,0-Z)

C...SWITCHES: REHUHB=100:10.FORMAT=900:10
C...SWITCHES:

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C FVAL
C

C***************************************************************************
EXTERNAL FVAL
DIMENSION BOUNDS(*) ,WS(1),WS1(1),STORE(47),STORE1(47)
INTEGER DIGITS
DOUBLE PRECISION SUM1,SUM2
DATA EPS2,EPS3,DIGITS/1.E-6,1.E-7,6/
DATA ZERO/0.0/,TWO/2.0/,EPS1/0.2/,ISIZE/47/,LIMIT/10/
DATA ASIZE/47.0/
C
C.. INITIALIZE
C
XOLD = 0.0
K = 0
KD = 0
FI = 0
H = 0
IVOLD = 0.0
SUM2 = 0.0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SUM1 = 0.0
XOLD = 0.0
HR = 0
FOLD = 0.0
C
C = B + DABS(B)
DO 100 M = 1, ISIZE
STORE1(M) = C
100 CONTINUE
NRoot = 0
J = 0
C
BOUNDS(1) = A
XSTART = A + (B-A)*0.2318
BOUNDS(2) = XSTART
XEND = B
I = 1
110 CONTINUE
N = NQ
C = XEND
IRET = 3
ARG = XSTART
GO TO 300
120 CONTINUE
FX = RES
IRET = 2
ARG = XEND
GO TO 300
130 CONTINUE
FD = RES
FOLD = FD
C
SUM1 = (FX+FD)/TWO
SUM2 = (FX*FX+FD*FD)/TWO
C
XOLD = 1080.0
XVOLD = 0.0
140 CONTINUE
BR = N - 1
XOLD = XSTART
H = (XEND-XSTART)/FLOAT(N)
K = 1
KD = 0
150 CONTINUE
IF (K .GT. BR) GO TO 180
C
XOLD = XOLD + H
IF (X.NE.NQ .AND. KD.EQ.1) GO TO 170
IRET = 1
ARG = XOLD
GO TO 300
C
160 CONTINUE
FD = RES
IF (FX*FD .LE. ZERO) GO TO 200
SUM1 = SUM1 + FD
SUM2 = SUM2 + FD*FD
170 CONTINUE
K = K + 1
KD = KD + 1
IF (KD .EQ. 2) KD = 0
GO TO 160
180 CONTINUE
IF (FOLD*FX .LE. ZERO) GO TO 200
AK = DFLAT(K)
XMEAN2 = (SUM1/AK)**2
XVAR = SUM2/AK - XMEAN2
C
IF (DABS(XMOLD-XVAR-XMEAN2) .LT. EPS1*DABSCXMEAH2*XMOLD))
  I  GO TO 190
C
XMOLD = XMEAN2
XMOLD = XVAR
N = 2*H
GO TO 140
190 CONTINUE
IF (XMEAN2 .GT. T2*XVAR) GO TO 220
IF (I + 5 .GT. L) L = 0
IF (L .EQ. 0) GO TO 260
I = I + 2
BOUNDS(I) = XSTART
BOUNDS(I+1) = (XEND+XSTART)/TWO
I = I + 2
BOUNDS(I) = BOUNDS(I-1)
BOUNDS(I+1) = XEND
GO TO 220
C
200 CONTINUE
IF (I + 7 .GT. L) L = 0
IF (L .EQ. 0) GO TO 260
I = I + 2
BOUNDS(I) = XSTART
C
IF (K .NE.1 .AND. K.LT.9) XOLD = XMOLD - H
BOUNDS(I-1) = XOLD
BOUNDS(I) = XOLD
I = I + 2
IF (K .EQ.1 .OR. K.GE.9) GO TO 210
BOUNDS(I-1) = XOLD + H
BOUNDS(I) = XOLD + H
I = I + 2
210 CONTINUE
BOUNDS(I-1) = XEND
220 CONTINUE
IF (I .LE. 0) GO TO 260
C
XSTART = BOUNDS(I)
XEND = BOUNDS(I+1)
I = I - 2
XMEAN = DABS(XEND-XSTART)/TWO
C
IF (XMEAN .LE. E) XMEAN = 1.0
IF (XEND - XSTART .LT. E*XMEAN) GO TO 230
GO TO 110
C
230 CONTINUE
IF (I + 2 .GT. L) L = 0
IF (L .EQ. 0) GO TO 260
C
J = J + 1
WS(J) = XSTART
IRET = 4
ARG = XSTART
GO TO 300
549

C

240 CONTINUE
   WSI(J) = RES
   J = J + 1
   WS(J) = XEND
   IRET = 6
   ARG = XEND
   GO TO 300
260 CONTINUE
   WSI(J) = RES
   GO TO 320

C

260 CONTINUE
   IF (J .EQ. 0) GO TO 350
C

270 CONTINUE
   XSTART = WS(J-1)
   IF (J .EQ. 2) GO TO 290
C

280 CONTINUE
   XMEAN = DABS(WS(J-3))
   IF (XMEAN .LT. 1.0) XMEAN = 1.0

   IF (DABS(WS(J-3)-WS(J)) .GT. EPS2*XMEAN) GO TO 290
   IF (WS1(J)*WS1(J-1) .LT. ZERO .AND. WS1(J-2)*WS1(J-3) .LT. ZERO)
      1   GO TO 290
C
   J = J - 2
   IF (J .GE. 4) GO TO 280
C

290 CONTINUE
   HROOT = HROOT + 1
   BOUNDS(HROOT) = XSTART
   HROOT = HROOT + 1
   BOUNDS(HROOT) = WS(J)
   J = J + 2
   IF (J .GE. 2) GO TO 270
   GO TO 380
300 CONTINUE
   HRE = 0
   ARGL = DABS(ARG)
   LLL = DIGITS - INT(DLOG10(ARG1+EPS3)+0.9)
   ARG1 = ARGL*10.0**LLL + 0.6
   ANASH = DMOD(ANASH,ASIZE)
   BNASH = INT(BNASH) + 1
   ARGL = DABS(ARG)
   IF (ARG1 .LE. 1.0) ARG1 = 1.0
310 CONTINUE
   IF (DABS(STORE1(HASH)-ARG) .LE. ARG1*EPS2) GO TO 330
   IF (STORE1(HASH) .LE. C) GO TO 320
   STORE1(HASH) = ARG
   RES = FVAL(ARG)
   STORE(HASH) = RES
   GO TO (160,130,120,240,250) IRET
C
320 CONTINUE
   HRE = HRE + 1
   IF (HRE .GT. LIMIT) GO TO 360
C
   HASH = NHASH + HRE*NRE
   NHASH = MOD(NHASH,ISIZE) + 1
GO TO 310

C 330 CONTINUE
RES = STORE(WHASH)
GO TO (160,130,120,240,250) IRET
C
C 340 CONTINUE
RES = FVAL(ARG)
GO TO (160,130,120,240,250) IRET
C
C 350 CONTINUE
HROOT = HRDOT/2
RETURN
EHD
C
*/
C INCLUDE(PROCESS)
FUNCTION FVAL(X)
IMPLICIT REAL*8(A-H,0-Z)
C
C .. SWITCHES: REUMB=100:10,FORMAT=900:10
C .. SWITCHES:
C***********************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C***********************************************************************
C COMMON /OTTO/ AC, BC, FK1, FK2, SIG1H, SIG2H, SIG3H, FCP, AAA, BBB, FT, IFG
PI = 3.141592654
C
COS3TH = 0.0
FVAL = 0.0
AJ1 = 0.0
AJ2 = 0.0
AJ3 = 0.0
IF (IFG .EQ. 1) THEN
AJ1 = SIG1H + SIG2H + X
AJ2 = (((SIG1H-SIG2H)**2+(SIG2H-X)**2+(X-SIG1H)**2)/6.0
AJ3 = ((SIG1H-AJ1/3.)*((SIG2H-AJ1/3.)*X-AJ1/3.))
ELSE IF (IFG .EQ. 2) THEN
RATIO1 = SIG1H/SIG3H
RATIO2 = SIG2H/SIG3H
AJ1 = RATIO1*X + RATIO2*X + X
AJ2 = (((RATIO1-RATIO2)*X)**2+(RATIO2*X-X)**2+(X-RATIO1*X)**2)
1 /6.0
AJ3 = (RATIO1*X-AJ1/3.)*(RATIO2*X-AJ1/3.)*(X-AJ1/3.)
ENDIF
C $ 
C IF (AJ2 .GE. (-0.001) .AND. AJ2 .LE. 0.001) AJ2 = 0.001
C C COS3TH = (3.0+DSQRT(3.0000)/2.0)*-(AJ3/DSQRT(AJ2 AJ2 AJ2))
C IF (COS3TH .GE. 0.0) THEN
ALAMB = FK1*DCOS((1/3.)*DA COS(FK2+COS3TH))
ELSE
ALAMB = FK1*DCOS((1/3.-FK2*COS3TH))
ENDIF
C FVAL = AC*AJ2/(FCP*FCP) + ALAMB*DSQRT(AJ2)/FCP + BC*AJ1/FCP - 1.0

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C
RETURN
END

C******************************************************************************
C INCLUDE(PROCESS)
SUBROUTINE FIDEX(FVALUE,FCP,PROPER,SIG1H,SIG2H,SIG3H,TOUT)
C IMPLICIT REAL*8(A-H,O-Z)
C . . .SWITCHES: RESUME=100:10,FORMAT=900:10
C . . .SWITCHES:
C******************************************************************************
C C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C******************************************************************************
DIMENSION PROPER(26)
C THIS SUBROUTINE CHECKS TO SEE IF THE FAILURE ENVELOPE HAS BEEN
C VIOLATED IN THE CURRENT ITERATION.
C FVALUE LE 0.0 NO FAILURE
C FVALUE = 0.0 JUST FAILED
C FVALUE GE 0.0 VIOLATED ENVELOPE
C
FVALUE = 0.0
COS3TH = 0.0
PI = 3.141592654
FCP=PROPER(5)
AC = PROPER(18)
BC = PROPER(19)
FK1 = PROPER(20)
FK2 = PROPER(21)

A11 = SIG1H + SIG2H + SIG3H
A12 = ((SIG1H-SIG2H)**2+(SIG2H-SIG3H)**2+(SIG3H-SIG1H)**2)/6.0
C IF (A12.GE.(-0.001) .AND. A12.LE.0.001) A12 = 0.001
COS3TH = (3.*DSQRT(3.0000)/2.0)*((A13/A12)**1.6)
C IF (COS3TH .GE. 0.0) THEN
ALAMB = FK1*DCOS((1/3.)*DACOS(FK2+COS3TH))
ELSE
ALAMB = FK1*DCOS(PI/3.-(1/3.)*DACOS((-FK2+COS3TH)))
ENDIF
C FVALUE=AC*A12/(FCP*FCP)+ALAMB*DSQRT(A12)/FCP+BC*A11/FCP-1.0
C RETURN
END

C******************************************************************************
C INCLUDE(PROCESS)
SUBROUTINE PSTSTB(STEB,PVAL,PDIR,IFG,TOUT)
IMPLICIT REAL*8(A-H,O-Z)
C . . .SWITCHES: RESUME=100:10,FORMAT=900:10
C . . .SWITCHES:
C******************************************************************************
C
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DEVCSF
C
C*******************************************************************************
C DIMENSION SSTEN(6),PVAL(3),PDIR(3,3),A(3,3),EVAL(3),EVEC(3,3)
C
C THIS SUBROUTINE DETERMINES THE PRINCIPAL STRESSES AND STRAINS.
C
C IFG=1...STRAIN AND IFG=2...STRESS
C
DO 100 IMN = 1, 3
   PDIR(IMN,1) = 0.0
   A(IMN,1) = 0.0
   EVEC(IMN,1) = 0.0
   PDIR(IMN,2) = 0.0
   A(IMN,2) = 0.0
   EVEC(IMN,2) = 0.0
   PDIR(IMN,3) = 0.0
   A(IMN,3) = 0.0
   EVEC(IMN,3) = 0.0
   PVAL(IMN) = 0.0
   EVAL(IMN) = 0.0
100 CONTINUE
PE1 = SSTEN(1)
PE2 = SSTEN(2)
PE3 = SSTEN(3)
PE4 = SSTEN(4)
PE5 = SSTEN(5)
PE6 = SSTEN(6)
A(1,1) = PE1
A(2,2) = PE2
A(3,3) = PE3
IF (IFG .EQ. 1) THEN
   A(1,2) = PE4/2.0
   A(2,1) = PE4/2.0
   A(1,3) = PE6/2.0
   A(3,1) = PE6/2.0
   A(2,3) = PE5/2.0
   A(3,2) = PE5/2.0
ELSE
   A(1,2) = PE4
   A(2,1) = PE4
   A(1,3) = PE6
   A(3,1) = PE6
   A(2,3) = PE5
   A(3,2) = PE5
ENDIF

C CALL IMSL ROUTINE FOR PRINCIPAL VALUES.
C
CALL DEVCSF (3, A, 3, EVAL, EVEC, 3)

IM = 3
P = EVEC(1,IM)
Q = EVEC(2,IM)
R = EVEC(3,IM)
SQ = DSGRT(P*P+Q*Q+R*R)
PDIR(1,1) = P/SQ
PDIR(1,2) = Q/SQ
PDIR(1,3) = R/SQ

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
PVAL(1) = EVAL(IM)
IM = 2
P = EVEC(1, IM)
Q = EVEC(2, IM)
R = EVEC(3, IM)
SQ = DSQRT(P*P + Q*Q + R*R)
PDIR(2, 1) = P/SQ
PDIR(2, 2) = Q/SQ
PDIR(2, 3) = R/SQ
PVAL(2) = EVAL(IM)
IM = 1
P = EVEC(1, IM)
Q = EVEC(2, IM)
R = EVEC(3, IM)
SQ = DSQRT(P*P + Q*Q + R*R)
PDIR(3, 1) = P/SQ
PDIR(3, 2) = Q/SQ
PDIR(3, 3) = R/SQ
PVAL(3) = EVAL(IM)

DO KM = 1, 3
  DO KW = 1, 3
    IF (DABS(PDIR(KM, KW)) .LE. 0.02) PDIR(KM, KW) = 0.0
  END DO
END DO

C...SWITCHES: REHUB=100:10,FORMAT=900:10

C...SWITCHES:

C SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT

C INCLUDE(PROCESS)
SUBROUTINE DTRANS(D, PDIR, IOUT)
IMPLICIT REAL*8(A-H, 0-Z)

C...SWITCHES: REHUB=100:10,FORMAT=900:10
C...SWITCHES:

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT

RETURN
END

C

--- END OF FILE ---
SUBROUTINE TO ROTATE THE MATERIAL CONSTITUTIVE MATRIX TO THE GLOBAL AXES GIVEN THE DIRECTION COSINES. THE ROTATED MATRIX WILL BE RETURNED IN ARRAY 'D'.

GENERATE THE TRANSFORMATION MATRIX 'T'

AL1 = PDIR(1,1)
AM1 = PDIR(1,2)
AN1 = PDIR(1,3)
AL2 = PDIR(2,1)
AM2 = PDIR(2,2)
AN2 = PDIR(2,3)
AL3 = PDIR(3,1)
AM3 = PDIR(3,2)
AN3 = PDIR(3,3)

IF (DABS(AL1) .LE. 0.01) AL1 = 0.0
IF (DABS(AM1) .LE. 0.01) AM1 = 0.0
IF (DABS(AN1) .LE. 0.01) AN1 = 0.0
IF (DABS(AL2) .LE. 0.01) AL2 = 0.0
IF (DABS(AM2) .LE. 0.01) AM2 = 0.0
IF (DABS(AN2) .LE. 0.01) AN2 = 0.0
IF (DABS(AL3) .LE. 0.01) AL3 = 0.0
IF (DABS(AM3) .LE. 0.01) AM3 = 0.0
IF (DABS(AN3) .LE. 0.01) AN3 = 0.0

T(1,1) = AL1*AL1
T(1,2) = AM1*AM1
T(1,3) = AN1*AN1
T(1,4) = AL1*AM1
T(1,5) = AM1*AN1
T(1,6) = AN1*AL1
T(2,1) = AL2*AL2
T(2,2) = AM2*AM2
T(2,3) = AN2*AN2
T(2,4) = AL2*AM2
T(2,5) = AM2*AN2
T(2,6) = AN2*AL2
T(3,1) = AL3*AL3
T(3,2) = AM3*AM3
T(3,3) = AN3*AN3
T(3,4) = AL3*AM3
T(3,5) = AM3*AN3
T(3,6) = AN3*AL3
T(4,1) = 2.0*AL1*AL2
T(4,2) = 2.0*AM1*AM2
T(4,3) = 2.0*AN1*AN2
T(4,4) = AL1*AM2 + AL2*AM1
T(4,5) = AM1*AN2 + AM2*AN1
T(4,6) = AM1*AL2 + AM2*AL1
T(5,1) = 2.0*AL2*AL3
T(5,2) = 2.0*AM2*AM3
T(5,3) = 2.0*AN2*AN3
T(5,4) = AL2*AM3 + AL3*AM2
T(5,5) = AM2*AN3 + AM3*AN2
T(5,6) = AN2*AL3 + AN3*AL2

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ T(5,6) = AN1*AL2 + AN2*AL3 \]
\[ T(6,1) = 2.0*AL1*AL3 \]
\[ T(6,2) = 2.0*AH1*AM1 \]
\[ T(6,3) = 2.0*AH1*AM3 \]
\[ T(6,4) = AL1*AM1 + AL3*AM3 \]
\[ T(6,5) = AH1*AM1 + AH3*AM3 \]

\[
D(GLOBAL) = T(TRANPOSE) * D(MATERIAL AXES) * T
\]

```
C  ... PERFORM D*T=DT
C
DO 120 I = 1, 6
   DO 110 J = 1, 6
      DT(I,J) = 0.0
      DO 100 K = 1, 6
         DT(I,J) = DT(I,J) + D(I,K) * T(K,J)
      100 CONTINUE
   110 CONTINUE
120 CONTINUE
C
C  ... PERFORM D(IXY)=T TRANS*DT
C
DO 150 I = 1, 6
   DO 140 J = 1, 6
      D(I,J) = 0.0
      DO 130 K = 1, 6
         D(I,J) = D(I,J) + T(K,I) * D(T,K,J)
      130 CONTINUE
   140 CONTINUE
150 CONTINUE
C
C   RETURN
END
```

```
C*************************************************************
C  INCLUDE(PROCESS)
C*************************************************************
C
SUBROUTINE CHKSTL(PROPER, TOLER, HSTIFF, CONVER, IELEM, L, ITYPE
C
$  \$
IMPLICIT REAL*8(A-H,0-Z)
C
C...SWITCHES: RESEMBLE=100:10, FORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C IOGET
C
C*************************************************************
```
SUBROUTINE TO CHECK CONVERGENCE AT EACH GAUSS POINT
OF STEEL LAYER 'J' OF ELEMENT 'I'. IF CONVERGENCE IS
NOT ACHIEVED THE CONSTITUTIVE MATRIX FOR THE POINT
AND THE STIFFNESS MATRIX FOR THE ELEMENT WILL BE
REGENERATED AND MORE ITERATIONS WILL BE CARRIED OUT.

DETERMINE THE STRAIN AT YIELD.

EPSNEW = 0.0
EPSNUE = 0.0
J = LNUM
CALL IOGET (LDEV1, 96, '(A96)', 5)
EPSX = STRAIN1(1)
EPSY = STRAIN1(2)
EPSZ = STRAIN1(5)
GAMAY = STRAIN1(4)
GAMAYZ = STRAIN1(5)
GAMAXZ = STRAIN1(6)

FOR EACH INTEGRATION POINT OF EACH LAYER AND ELEMENT RECOVER THE
STRAINS AND PROCEED WITH THE CONVERGENCE CHECK

SIGYLD = HISTO(L,13,J)
ESYLD = HISTO(L,14,J)
EPSYLD = HISTO(L,12,J)

REDUCE TRUECAUTION ERRORS ...

XYZ = MAX1(MAX1(MAX1(MAX1(MAX1(MAX1(DABS(EPSX),DABS(EPSY)),DABS(EPSY)),DABS(GAMAY)),DABS(GAMAYZ)),DABS(GAMAXZ)),DABS(EPSZ))
IF (DABS(EPSX).LT.XYZ*10.0E-7) EPSX = 0.0
IF (DABS(EPSY).LT.XYZ*10.0E-7) EPSY = 0.0
IF (DABS(EPSZ).LT.XYZ*10.0E-7) EPSZ = 0.0
IF (DABS(GAMAY).LT.XYZ*10.0E-7) GAMAY = 0.0
IF (DABS(GAMAYZ).LT.XYZ*10.0E-7) GAMAYZ = 0.0
IF (DABS(GAMAXZ).LT.XYZ*10.0E-7) GAMAXZ = 0.0

CALCULATE THE CURRENT STRAIN IN THE DIRECTION OF REBARS

AL1 = HISTO(L,17,J)
AM1 = HISTO(L,18,J)
AB1 = HISTO(L,19,J)
EPSSTL = EPSX*AL1+AL1 + EPSY*AM1+AM1 + EPSZ*AB1+AB1 + GAMAY*AL1+AL1+1 + GAMAYZ*AM1+AM1 + GAMAXZ*AB1+AB1
IF (DABS(EPSSTL) .LE. 1.0E-7) EPSSTL = 0.0

COMPARE THIS WITH THE PREVIOUS STRAIN.

EPSOLD = HISTO(L,10,J)
IF (DABS(EPSSTL) .GE. .999*DABS(EPSOLD)) THEN

CHANGE THE STATE OF THE POINT TO LOADING


* LOADING *

---
CONSIDER POSSIBILITY OF UNLOADING IN THE NEXT STEPS. THEREFORE, SAVE THE STRAIN FROM WHICH UNLOADING MIGHT INITIATE.

\[
\text{HISTO}(L,10,J) = \text{EPSSTL}
\]

SAVE THE CURRENT STRAIN.

\[
\text{HISTO}(L,11,J) = \text{EPSSTL}
\]

COMPARE THE CURRENT STRAIN WITH THE YIELD STRAIN.

IF (DABS(EPSSTL) .LE. DABS(EPSYLD)) RETURN

POST-YIELD CALCULATIONS

\[
\text{HISTO}(L,16,J) = 2.
\]

DETERMINE SECAN STRESS'S MODULUS CORRESPONDING TO EPSST

\[
\text{ALFA} = \text{HARDENING PARAMETER}
\]

\[
\text{ALFA} = \text{PROPER}(6)
\]

\[
\text{EIBIT} = \text{EIBIT}(L,14,J)
\]

\[
\text{SIGULT} = \text{SIGYLD}(1.0-\text{ALFA}) + \text{ALFA} \cdot \text{EIBIT} \cdot \text{DABS} \cdot \text{EPSULT}
\]

TOTAL AREA UNDER STRESS-STRAIN CURVE

\[
\text{AREA} = 0.5 \cdot \text{SIGYLD} \cdot \text{DABS} \cdot \text{SIGYLD} + (\text{DABS} \cdot \text{EPSYLD} \cdot \text{SIGYLD} + (\text{SIGULT} \cdot \text{SIGYLD}) \cdot (\text{DABS} \cdot \text{EPSYLD} \cdot \text{SIGYLD})) \cdot 0.50
\]

FIND THE STIFFNESS CORRESPONDING TO THE UPDATED STRAIN

\[
\text{ES} = (\text{SIGYLD} \cdot (1.0-\text{ALFA}) + \text{ALFA} \cdot \text{EIBIT} \cdot \text{DABS} \cdot \text{EPSSTL}) / \text{DABS} \cdot \text{EPSSTL}
\]

\[
\text{SUBARA} = 0.5 \cdot (\text{HISTO}(L,1,1) \cdot \text{DABS}(\text{EPSOLD}) + \text{ES} \cdot \text{DABS}(\text{EPSSTL}) \cdot (\text{DABS}(\text{EPSSTL}) \cdot \text{DABS}(\text{EPSOLD}) \cdot \text{SIGYLD} + (\text{SIGYLD} \cdot \text{SIGYLD}) \cdot (\text{DABS}(\text{EPSYLD}) \cdot \text{HISTO}(L,1,7))
\]

IFLAG = 1

\[
\text{PERCST} = (\text{SUBARA} / \text{AREA}) \cdot 100
\]

IF (PERCST .GT. TOLER) THEN

\[
\text{ESNEW} = \text{ES}
\]

IFLAG = 0

ENDIF

\[
\text{TOLERE} = 1.0
\]

SIGCUR = HISTO(L,1,1) \cdot DABS(\text{EPSSTL})

IF (SIGCUR .GT. SIGYLD .AND. FLAG.EQ.1) THEN

\[
\text{AREA} = 0.5 \cdot \text{SIGYLD} \cdot \text{DABS}(\text{EPSYLD}) + (\text{DABS}(\text{EPSYLD}) \cdot \text{SIGYLD} + (\text{SIGYLD} \cdot \text{SIGYLD}) \cdot (\text{DABS}(\text{EPSYLD}) \cdot \text{HISTO}(L,1,3))
\]

\[
\text{TOLERE}/100.0 = 0.5 \cdot \text{DABS}(\text{EPSOLD}) \cdot \text{DABS}(\text{EPSYLD}) \cdot 0.5 + \text{AREA} + \text{TOLERE}/100.0 - 0.5 \cdot \text{DABS}(\text{EPSOLD}) \cdot \text{DABS}(\text{EPSOLD}) \cdot \text{HISTO}(L,1,3)
\]
LENGTH CORRECT FOR SMALL ALFA'S

\[
\text{LENGTH} = \sqrt{\text{SIGYLD}^2 + \text{EPSYLD}^2}
\]

\[
\text{ALTUDE} = \frac{\text{AREA}}{(\text{LENGTH} \times 0.5)}
\]

\[
\text{EPSNEW} = \text{ALTUDE} + \text{EPSYLD}
\]

\[
\text{ESNEW} = \frac{\text{SIGYLD} \times (1 - \text{ALFA}) + \text{ALFA} \times \text{EINIT} \times \text{DABS(ESNEW))}}{\text{DABS(EPShew)}}
\]

IFLAG = 0

\[
\text{HISTO}(L,10,J) = \text{EPSNEW}
\]

\[
\text{HISTO}(L,11,J) = \text{EPSNEW}
\]

ENDIF

IF (PERCNT.GT.TOLER .OR. IFLAG.EQ.0) THEN

IN ORDER TO UPDATE THE CONSTITUTIVE LAW AT THE POINT

SIMPLY DIVIDE THE PREVIOUSLY ROTATED CONSTITUTIVE

MATRIX STORED IN ARRAY 'CRACK' BY THE OLD 'E' AND

THEN MULTIPLY IT BY THE NEW SECANT 'E'.

\[
\text{RATIO} = \frac{\text{ESHEW}}{\text{HISTO}(L,1,J)}
\]

DO 100 K = 2, 37

\[
\text{CRACK}(L,K,J) = \text{CRACK}(L,K,J) \times \text{RATIO}
\]

CONTINUE

ENDIF

STORE THE CURRENT SECANT YOUNG'S MODULUS.

\[
\text{HISTO}(L,1,J) = \text{ESNEW}
\]

UPDATE ELEMENT STIFFNESS MATRIX AND CARRY OUT FURTHER

ITERATIONS.

\[
\text{BSTIFF} = 1
\]

\[
\text{CONVER} = 999.
\]

\[
\text{CHGUS} = \text{TRUE}.
\]

ENDIF

ENDIF

RETURN

ENDIF

--------

* UNLOADING *

--------

IF (DABS(EPStl) .LT. .999*DABS(EPSold)) THEN

GET THE STRAIN IN PREVIOUS ITERATIONS.

\[
\text{PREEPS} = \text{HISTO}(L,11,J)
\]

\[
\text{VAR} = \text{EPSStl} \times \text{PREEPS}
\]

IF (VAR .LT. 0.0) THEN

\[
\text{HISTO}(L,10,J) = 0.0
\]

\[
\text{HISTO}(L,11,J) = 0.0
\]

\[
\text{HISTO}(L,16,J) = 1.0
\]

\[
\text{CONVER} = 999.0
\]

RETURN

ENDIF
IF (ABS(EPSSTL) .LT. .999*ABS(PREEPS)) THEN

SET THE STATE OF THE POINT TO UNLOADING

HISTO(L,16,J) = 2.

SAVE THE CURRENT STRAINS

HISTO(L,11,J) = EPSSTL

RETURN

ENDIF

-- RELOADING --

IF (ABS(EPSSTL) .GE. .999*ABS(PREEPS)) THEN

HISTO(L,16,J) = 3.

SAVE THE CURRENT STRAINS

HISTO(L,11,J) = EPSSTL

ENDIF

ENDIF

RETURN

END

C
C __________________ S I M U L __________________
C
C INCLUDE(PROCESS)
C FUNCTION SIMUL(N,A, EPS, NRC, IOUT)
IMPLICIT REAL*8(A-H,O-Z)
C
C SWITCHES: REENUM=100:10,FORMAT=900:10
C SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C DIMENSION IR0W(40), JCOL(40), JORD(40), T4(40), & (NRC,*)
C REAL*12 A, AIJCK, PIVOT, EPS
C
C FUNCTION TO COMPUTE THE INVERSE OF THE N BY N
C POTENTIAL PIVOT OF LARGEST MAGNITUDE BE SMALLER
C IN MAGNITUDE THAN EPS, THE MATRIX IS CONSIDERED
C TO BE SINGULAR AND A TRUE ZERO IS RETURNED AS
C THE VALUE OF THE FUNCTION.
C
C MAX = N

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
DO 100 M = 1, 40
   IROW(H) = 0
   JCOL(M) = 0
   JORD(M) = 0
   Y(M) = 0.0
100 CONTINUE

ELIMINATE ANY SMALL ELEMENT WHICH CAN CAUSE
OVERFLOW PROBLEM DURING MATRIX INVERSION.

............ SEARCH FOR THE LARGEST ELEMENT

BIG = 0.0
DO 120 I = 1, N
   BIG = MAX1(DABS(A(I,J)),BIG)
120 CONTINUE

VERYSH = BIG*10.**(-10)
DO 140 I = 1, N
   IF (DABS(A(I,J)) .LE. VERYSH) A(I,J) = 0.0
140 CONTINUE

... IS N LARGER THAN 40

IF (N .LE. 40) GO TO 160
WRITE (*, 900) N
SIMUL = 0.0
RETURN

BEGIN ELIMINATION PROCEDURE

DO 270 K = 1, H
   KHM = K - 1
   PIVOT = 0.0
   DO 210 I = 1, H
      J = 1
      SCAN IROW AND JCOL ARRAYS FOR INVALID PIVOT SUBSCRIPTS
      IF (I .EQ. 1) GO TO 180
      IF (I .EQ. IROW(SCAN)) GO TO 190
      IF (J .EQ. JCOL(SCAN)) GO TO 190
      CONTINUE
180 CONTINUE

SEARCH FOR THE PIVOT ELEMENT

DO 200 J = 1, H
   IF (KHM .EQ. 1) GO TO 180
   IF (I .EQ. IROW(SCAN)) GO TO 190
   IF (J .EQ. JCOL(SCAN)) GO TO 190
   CONTINUE
190 CONTINUE

CONTINUE
200 CONTINUE
210 CONTINUE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C
C INSURE THAT SELECTED PIVOT IS LARGER THAN EPS
C
IF (DABS(PIVOT) .GT. EPS) GO TO 220
SIMUL = 0.0
RETURN
C
220 CONTINUE
UPDATE THE DETERMINANT VALUE
C
IROWK = IROW(K)
JCOLK = JCOL(K)
DETER = DETER * PIVOT
C
NORMALIZE PIVOT ROW ELEMENTS
C
DO 230 J = 1, MAX
A(IROWK,J) = A(IROWK,J)/PIVOT
CONTINUE
C
CARRY OUT ELIMINATION AND DEVELOPE INVERSE
C
A(IROWK,JCOLK) = 1./PIVOT
DO 260 I = 1, N
AIJCK = A(I,JCOLK)
IF (I .EQ. IROWK) GO TO 260
A(I,JCOLK) = -AIJCK/PIVOT
DO 240 J = 1, MAX
IF (J .NE. JCOLK) A(I,J) = A(I,J) - AIJCK*A(IROWK,J)
CONTINUE
CONTINUE
CONTINUE
C
ORDER SOLUTION VALUES (IF ANY) AND CREATE JORD ARRAY
C
DO 280 I = 1, N
IROWI = IROW(I)
JCOLI = JCOL(I)
JORD(IROWI) = JCOLI
CONTINUE
C
ADJUST SIGN OF DETERMINANT
C
IHTCH = 0
ENH = N - 1
DO 310 I = 1, ENH
JP1 = I + 1
DO 300 J = JP1, N
IF (JORD(J) .GE. JORD(I)) GO TO 290
JTEMP = JORD(J)
JORD(J) = JORD(I)
JORD(I) = JTEMP
IHTCH = IHTCH + 1
CONTINUE
CONTINUE
CONTINUE
CONTINUE
IF (IHTCH/2*2 .NE. IHTCH) DETER = -DETER
C
C IF INDIC IS NEGATIVE OR ZERO, UNSCRAMBLE THE INVERSE
C .... FIRST BY ROWS
C
320 CONTINUE
    DO 360 J = 1, N
    DO 330 I = 1, N
      IROWI = IROW(I)
      JCOLI = JCOL(I)
      Y(JCOLI) = A(IROWI, J)
    330     CONTINUE
    DO 360 J = 1, N
      A(I, J) = Y(J)
  360    CONTINUE
  380 CONTINUE
C
C       THEN BY COLUMNS
C
    DO 380 I = 1, N
    DO 360 J = 1, N
      IROWJ = IROW(J)
      JCOLJ = JCOL(J)
      Y(IROWJ) = A(I, JCOLJ)
  360    CONTINUE
    DO 380 J = 1, N
      A(I, J) = Y(J)
  370    CONTINUE
  380 CONTINUE
C
C    RETURN FOR INDIC NEGATIVE OR ZERO
C
C      SIMUL = DETER
    RETURN
    FORMAT FOR OUTPUT STATEMENT
C
  900 FORMAT(/,1X,'ERROR IN SUBROUTINE SIMUL',/,'Y=',13,1X,' WHICH EXCEEDS ITS MAX. VALUE OF 40')
    END
C
C==================================SOLVE================================
C
C INCLUDE (PROCESS)
C SUBROUTINE SOLVE(A,C,B,DIAG,BEQ,AFAC,BACK)
C
C================================================================================
C
C PROGRAM:  
C PROGRAM 'SOLVE' IS USED TO SOLVE A SERIES OF BARRDED
C EIGENSYMMETRIC LINEAR EQUATIONS USING THE GAUSS ELIMINATION/BACK
C SUBSTITUTION WITH NO COLUMN PIVOTING.
C
C STORAGE:  COEFFICIENT MATRIX SHOULD BE STORED IN TWO ONE
C DIMENSIONAL ARRAYS USING THE SKYLINE OR THE
C PROFILE METHOD
C A(K ) = UPPER TRIANGULAR MATRIX
C C(K ) = LOWER TRIANGULAR MATRIX
C H(K ) = RIGHT HAND SIDE VECTOR OR CALL
C V(K ) = VECTOR OF UNEVALUES OR RETURN
C
C================================================================================
C
IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: RESUMB=100:10,FORMAT=900:10
C...SWITCHES:
SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

C
C DOTPRO
C
C****************************************************************************
LOGICAL AFAC,BACK
DIMENSION A(1),C(1),B(1),JDIAG(1)
C
C FACTOR A TO UT*U*D*UT, REDUCE B TO Y
C
JR = 0
DO 140 J = 1, NEQ
JD = JDIAG(J)
JH = JD - JR
IF (JH .LE. 1) GO TO 130
IS = J + 1 - JH
IE = J - 1
IF (.NOT.AFAC) GO TO 120
K = JR + 1
ID = 0
C
C REDUCE ALL EQUATIONS EXCEPT DIAGNAL
C
DO 110 I = IS, IE
IR = ID
ID = JDIAG(I)
IH = MINO(ID-JR-1,1-IS)
IF (IH .EQ. 0) GO TO 100
A(K) = A(K) - DOTPRO(A(K-IH),C(ID-IH),IH)
C(K) = C(K) - DOTPRO(C(K-IH),A(ID-IH),IH)
100 CONTINUE
IF (A(ID) .NE. 0.0) C(K) = C(K)/A(ID)
K = K + 1
110 CONTINUE
C
C REDUCE THE DIAGNAL TERM
C
A(JD) = A(JD) - DOTPRO(A(JR+1),C(JR+1),JH-1)
C
C FORWARD REDUCE THE RIGHT HAND SIDE
C
120 CONTINUE
IF (BACK) B(J) = B(J) - DOTPRO(C(JR+1),B(IS),JH-1)
130 CONTINUE
JR = JD
140 CONTINUE
IF (.NOT.BACK) RETURN
C
C BACK SUBSTITUTION
C
J = NEQ
JD = JDIAG(J)
150 CONTINUE
IF (A(JD) .NE. 0.0) B(J) = B(J)/A(JD)
D = B(J)
J = J - 1
IF (J .LE. 0) RETURN
JR = JDIAG(J)
IF (JD - JR .LE. 1) GO TO 170
IS = J - JD + JR + 2
K = JR - IS + 1
C
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C
C ==.......................... SOLVE2 ...............................==
C
C INCLUDE (PROCESS)
C SUBROUTINE SOLVE2(A,R,JDIAG,HEQU,KKK,IOUT)
C
C ==.......................... SOLVE2 ...............................==
C
C THIS PROGRAM IS USED TO SOLVE FIBITE ELEMENT STATIC EQUILIB.
C EQUATIONS IN CORE, USING COMPACTED STORAGE AND COLUMN REDUCED
C SCHEME
C
C IMPLICIT REAL*8(A-H,Q-Z)
C...SWITCHES: RESUMB=100.10,POSMAT=900.10
C...SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C NO SUBROUTINES OR FUNCTION CALLED FROM THIS PROGRAM UNIT
C
C==.......................... DIMENSION A(1),JDIA(1),R(1) ==
C
C PERFORM L=D=L FACTORIZATION OF THE STIFFNESS MATRIX
C
C IF (KKK.LT.2) THEN
DO 140 K = 1, KKK
     KN = JDIA(K)
     KL = KN + 1
     KU = JDIA(K+1) - 1
     KH = KU - KL
     IF (KH.GT.0) THEN
          K = N - KH
          IC = 0
          KLT = KU
          DO 110 J = 1, KH
               IC = IC + 1
               KLT = KLT - 1
               KI = JDIA(K)
               KD = JDIA(K+1) - KI - 1
               IF (KD.GT.0) THEN
                    KK = MIN0(IC,KD)
                    C = 0.
                    DO 100 L = 1, KK
                         C = C + A(KL+L)*A(KLT-L)
                         100 CONTINUE
                    KLT = A(KLT) - C
               CONTINUE
          K = K + 1
     ELSE IF (KH.LT.0) THEN
          GO TO 130
     ENDIF
ENDIF

C
K = N
B = 0.
C*V D IR: IGNORE RECRDEPS
DO 120 KK = K L, KU
   K = K - 1
   K I = JDIAG(K)
   C = A(KK)/A(KI)
   B = B + C*A(KK)
   A(KK) = C
120  CONTINUE
A(KS) = A(KW) - B
CONTINUE
IF (DABS(A(KW)) .LE. 1.E-9) THEN
   WRITE (OUT, 900) N, A(KW)
   STOP
ENDIF
140  CONTINUE
RETURN
C
REDUCE THE RIGHT-HAND-SIDE LOAD VECTOR
C
ELSE
   DO 160 N = 1, NEQU
      KL = JDIAG(N) + 1
      KU = JDIAG(N + 1) - 1
      IF (KU - KL .GE. 0) THEN
         K = N
         C = 0.
         DO 150 KK = KL, KU
            K = K - 1
            C = C + A(KK)*R(K)
150        CONTINUE
         R(N) = R(N) - C
      ELSE
         DO 160 KK = KL, KU
            K = K - 1
            R(K) = R(K) - A(KK)*R(K)
160        CONTINUE
      ENDIF
160  CONTINUE
C
BACK-SUBSTITUTE
C
DO 170 N = 1, NEQU
   K = JDIAG(N)
   R(N) = R(N)/A(K)
170  CONTINUE
C
IF (NEQU .EQ. 1) RETURN
   N = NEQU
   DO 190 L = 2, NEQU
      KL = JDIAG(N) + 1
      KU = JDIAG(N + 1) - 1
      IF (KU - KL .GE. 0) THEN
         K = N
      ELSE
         DO 190 KK = KL, KU
            K = K - 1
            R(K) = R(K) - A(KK)*R(K)
190        CONTINUE
      ENDIF
190  CONTINUE
C*VDIR: IGNORE RECRDEPS
   DO 180 KK = KL, KU
      K = K - 1
      R(K) = R(K) - A(KK)*R(K)
180  CONTINUE
ENDIF
N = N - 1
CONTINUE
ENDIF
RETURN
900 FORMAT(//1X,'STOP - STIFFNESS MATRIX NOT POSITIVE DEFINITE')
FIXED PIVOT FOR EQUATION '144/1X,'PIVOT = ',E20.12)
END

C
SERVICE (PROCESS)
FUNCTION DOTP (A,B,N)
IMPLICIT REAL*8 (A-H,0-Z)
C.V. Switches: RENUM=100;10,FFORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C*************************************************************
REAL*8 A,B,DOTP,TEMP
DIMENSION A (1), B(1)
TEMP = 0.0
DO 100 I = 1, N
  TEMP = TEMP + A(I) * B(I)
100 CONTINUE
DOTP = TEMP

RETURN
END
C
C*************************************************************
C SUBROUTINE CROSS(V1,V2,V3)
IMPLICIT REAL*8 (A-H,0-Z)
C...SWITCHES: RENUM=100;10,FFORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C*************************************************************
DIMENSION V1(3),V2(3),V3(3)
V3(1) = V1(2) * V2(3) - V1(3) * V2(2)
V3(2) = V1(3) * V2(1) - V1(1) * V2(3)
V3(3) = V1(1) * V2(2) - V1(2) * V2(1)
RETURN
END
C
C*************************************************************
C SUBROUTINE UNITVIC(V1,N)
IMPLICIT REAL*8 (A-H,0-Z)
C...SWITCHES: RENUM=100;10,FFORMAT=900:10
C...SWITCHES:
C*************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C*************************************************************
DIMENSION V1(1)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SQSUM = 0.0D0
DO 100 K1 = 1, N
    SQSUM = SQSUM + V1(K1)**2
100 CONTINUE
VNORM = DSQRT(SQSUM)

IF (VNORM .NE. 0.0D0) THEN
    DO 110 K1 = 1, N
        V1(K1) = V1(K1)/VNORM
110 CONTINUE
ENDIF

RETURN
END

C SUBROUTINE UNITS(V1,N)

IMPLICIT REAL*8 (A-H,O-Z)
C ..SWITCHES: REHUMB=100:10,FORMAT=900:10
C ..SWITCHES:
C***...
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
REAL*4 VI
DIMENSION V1(1)
C
SQSUM = 0.0D0
DO 100 K1 = 1, N
    SQSUM = SQSUM + V1(K1)**2
100 CONTINUE
VNORM = DSQRT(SQSUM)

IF (VNORM .NE. 0.0D0) THEN
    DO 110 K1 = 1, N
        V1(K1) = V1(K1)/VNORM
110 CONTINUE
ENDIF

RETURN
END

C INCLUDE (PROCESS)

SUBROUTINE DIRVEC(V1,V2,V3)

========= DIRVEC ===============

PROGRAM:

DIRCOS EVALUATES THE DIRECTION COSINES OF THE Y.PRIME AND
THE X.PRIME AXES GIVEN THE Z.PRIME DIRECTION.

ARGUMENT LIST
c I V1 = vector containing the direction cosines of the x-prime
c I V2 = vector containing the direction cosines of the y-prime
c I V3 = vector containing the direction cosines of the z-prime

c implicit real*8 (a-h,o-z)
c...switches: renum=100:10,format=900:10

c...switches:
c
subroutines and functions called from this routine

c no subroutines or functions called from this program unit

c***********************************************************************
dimension v1(3),v2(3),v3(3)
c
--- third unit vector is user specified the other two are evaluated
c as described by comment lines

tx = v3(1)
ty = v3(2)
tz = v3(3)

--- the first vector is chosen to be (v3 x k)

cnorm = dsqrt(tx**2+ty**2)
if (dabs(cnorm) .lt. 1.0d-8) then
  v1(1) = 1.
  v1(2) = 0.
  v1(3) = 0.
else
  v1(1) = ty/cnorm
  v1(2) = -tx/cnorm
  v1(3) = 0.
endif

c --- the second vector is defined as (v3 x v1)

c
  v2 = ty*v1(3) - tz*v1(2)
  v3 = (-tx*v1(3)) + tz*v1(1)
  v2 = tx*v1(2) - ty*v1(1)
  cnorm = dsqrt(vx**2+vy**2+vz**2)
  v2(1) = vx/cnorm
  v2(2) = vy/cnorm
  v2(3) = vz/cnorm

c return
end

c include(process)

subroutine out28(idut)
implicit real*8 (a-h,o-z)
c...switches: renum=100:10,format=900:10

c...switches:
c
subroutines and functions called from this routine

c
C ELINFO ELBTH LYINFO ISH3DG ISHSHL
C ISHGD SBORM DIREC JACBD COORD GETBK
C JACSHL C
C
C*******************************************************************************
CHARACTER*8 ID1
REAL*8 LTHICK,SHLZ
INTEGER ELNUM,ELNMS
COMMON/OUTPT1/IOFLAG,IFCNAME
COMMON/ROCKS/HISTO(27,70,16),CRACK(27,280,16),HIHIST,HIHIST:
$ ,IHIST2
COMMON/SHELDIR/VECT(3,3,9)
COMMON/ABC/TESTF(27,80,16),ITBSPG,ITBSG1,ITBSG2
COMMON/LPRO/PROPER(26,16)
COMMON/INPUT/WIPZ,WIPETA,WIPS,WIP,IBTCD
COMMON/INPUT/MAITPE(10)
COMMON/LAYERS/LTHICK(9),2S(9),DCS(3,3),ELAYRS,MATRL,LYNUM
COMMON/PSHAPI/I2(20,27),W11(20,27),BETA(20,27),S1(20,27),S1(27)
COMMON/PSHAP/I2(27)
COMMON/INPUT/MODES,WELEM,NSDF,ELINC,MMIT,IFLAG1,IFLAG2,ITDIM
$ ,B-node,BCOLOR,FPFREE
COMMON/INCR11/FRAC(10),ELINC1(10),LOCISP,IBCPR
COMMON/MATER/DEP(6,6)
COMMON/OUTPT2/IDEL(8000),IDMOD(10000)
COMMON/CONTRIC/ICBREM,BIT
COMMON/INPUT/IP(20,8000)
COMMON/INPUT/IISPR(10000)
COMMON/PSK2/SHLZ(3,10000)
COMMON/INPUT/TRICK(9),IFLAG
COMMON/ICBREM/ICBREM,ICBREM
DIMENSION POIR(3,3),V3(3)

C OUTPUT MATERIAL INFORMATION AT SAMPLE POINT OF EACH CONCRETE
C /STEEL SAMPLE POINT. IT IS SET UP TO POST PROCESS CRACK INFO
C FOR GRAPHIC PROCESSING OF CARDS.
C
ICRR = 40
ILBE = 1
ICSTR = 10
ICKR = 72
INTFG = 0
IF (BTEST(IOFLAG,1).AND. BTEST(IOFLAG,2)) INTFG = 1
C
WRITE (ICRR, 1320) IECREM
WRITE (ICRR, 1340) IECREM
REWRIT IECREM
REWRIT ITBSPG
REWRIT ITBSG1
REWRIT IHIST
IF (ICBREM .EQ. 1) THEN
IF (MOD(IECREM,10).EQ. 0) THEN
   ID1 = 'CRACK_ORIENTATION'
   WRITE (ICKR, 'A6') ID1
   WRITE (ICRR, 'A6') ' 800'
ENDIF
WRITE (ICKR, 1430) IECREM
WRITE (ICRR, 1440) ID1
WRITE (ICRR, 'A6') ' 801'
WHITE (+, *) 'IFLAG1, IECREM, IECREM', IECREM
ENDIF
EENDIF

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
DO 260 ELNUM = 1, SELEN
CALL ELISFO (ELNUM, ITYPE, BBEL, IFLAG, ISTART, LIBES)
CALL ELISFM (ELNUM, IDENT, ITCOD, BIPXI, BIPETA, BIPS, 
1 MATBUH, THICK)
DO 230 LYBUM = 1, BLAYRS

C
D11 = 0.0
D12 = 0.0
D13 = 0.0
D21 = 0.0
D22 = 0.0
D23 = 0.0
D31 = 0.0
D32 = 0.0
D33 = 0.0
PERCBT = 0.0
C
CALL LYISFO (LYNUM, LTHICK, ZS, MATEL, DCS, BSEL, ELNUM, 
1 BIPXI, BIPETA, BIPS)
C
IF (MATYPE (MATNUM) .LE. 4) GO TO 240
BIP = BIPXI * BIPETA * BIPS
WRITE(*,*)'************ PROCESSING CONCRETE OUTPUT**********'
MM = (ELNUM-1)*27 + 1
READ (IHISTY) ELBUMB, LYBUMB, BGAUS, ((CRACK(L,K,LYBUM),K 
1 = 1,260), L = 1, HIP)
IF (ELBUMB .BE. ELBUM .OR. LYBUMB .BE. LYBUM .OR. BGAUS .BE. BIP 
1 ) THEN 
WRITE (*, *)'ERROR IN READING IHISTY IN OUT28'
WRITE (*, *)'ELNUM', ELNUM, 'ELNUM', ELNUM, 'LYNUM', 
1 LYBUM, 'LYNUM', 'LYNUM', 'LYNUM', 'BGAUS', 'BGAUS', 'BIP', 
2 BIP
STOP
ESDF
READ (ITHESPG) ELBUMB, LYBUMB, BGAUS, MATEL, ((THERSF(L,K, 
1 LTHICK), L = 1, HIP), ((ESTS(L,K,LYNUM),A 
2 1,700), L = 1, HIP), (PROPER(KM,MATER), KM = 1, 26)
IF (ELBUMB .BE. ELBUM .OR. LYBUMB .BE. LYBUM .OR. BGAUS .BE. BIP 
1 ) THEN 
WRITE (*, *)'ERROR IN READING ITHESPG IN OUT28'
WRITE (*, *)'ELNUM', ELNUM, 'ELNUM', ELNUM, 'LYNUM', 
1 LYBUM, 'LYNUM', 'LYNUM', 'BGAUS', 'BGAUS', 'BIP', 
2 BIP
STOP
ESDF
C
C
C.... TYPE OF MATERIAL 0=STEEL
C 1=CONCRETE
C 2=ELASTIC
C
IF (ITYPE .LE. 0) GO TO 240
IF (ITYPE .LE. 0 .OR. IDENT .LE. 0) THEN
IF (ITYPE .GT. 100) THEN
IF (IFLAG .EQ. 0) THEN
BCE = 3*BSEL
CALL ISE3DG (ITYPE, BSEL, ITYPE)
ELSE IF (IFLAG .EQ. 4) THEN
BCE = 6*BSEL
CALL ISE6SL (ITYPE, BSEL, ITYPE)
ESDF
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ELSE IF (ITYPE .GT. 200) THEN
    ECB = 2*BBEL
    CALL ISERUG (ITYPE, BBEL, IERROR)
    IF (IFLAG .EQ. 3) THEN
    ELSE
    ENDIF
ENDIF

C**** IF (ICFG0D .EQ. 0) THEN
IF (MOD(INCREM,ICNTR) .EQ. 0) THEN
    IF (IFLAG .EQ. 4) THEN
        DO 100 K1 = 1, BBEL
            KP = MOP(K1,ELNUM)
            ICODE = ISPB(KP,4)
            ⋮
            IF (ICODE .GT. 0) THEN
                CALL SHBORM (ELNUM, K1, BBEL, ITYPE,
                    1,VECT(1,1,K1),VECT(1,2,K1),VECT(1,3
                    ,K1))
            ELSE
                VECT(1,3,K1) = DBLE(SHELLZ(1,KP))
                VECT(2,3,K1) = DBLE(SHELLZ(2,KP))
                VECT(3,3,K1) = DBLE(SHELLZ(3,KP))
                CALL DIRVEC (VECTd , 1 ,K1) , VECTd .2 ,K1).
            ⋮
        100 CONTINUE
        ENDIF
ENDIF

C IF (ICELL .EQ. 0) THEN
    CALL SHBORM (ELNUM, X1, BBEL, ITYPE,
        1,VECT(1,1,K1),VECT(1,2,K1),VECT(1,3
        ,K1))
    ELSE
        VECT(1,3,K1) = DBLE(SHELLZ(1,KP))
        VECT(2,3,K1) = DBLE(SHELLZ(2,KP))
        VECT(3,3,K1) = DBLE(SHELLZ(3,KP))
        CALL DIRVEC (VECTd , 1 ,K1) , VECTd .2 ,K1).
    ⋮
ELSE
ENDIF

C IF (IPRL .EQ. 0) THEN
    IF (IPROPER .EQ. 0) THEN
        WRITE (ICRKN, 1010) LYBUM
    IF (IPROPER .EQ. 1) WRITE (ICRKN, 1020) LYBUM
    ⋮
ENDIF

C ⋮ STEEL LAYER OUTPUT THE RELEVANT STRESS/STRAIN(PRIB .), STIFFNESS
AND OTHER STATE VARIABLES
C IF (L .LE. 5) GO TO 200
WRITE (ICRKN, 900) HISTO(L,16,LYNUM)
WRITE (ICRKN, 910) HISTO(L,16,LYNUM)
WRITE (ICRKN, 920) HISTO(L,1,LYNUM)
WRITE (ICRKN, 930) HISTO(L,1,LYNUM)
SIGSTL = HISTO(L,1,LYNUM)+HISTO(L,1,LYNUM)
WRITE (ICRKN, 940) SIGSTL
WRITE (ICRKN, 960) HISTO(L,1,LYNUM)
WRITE (ICRKN, 960) HISTO(L,1,LYNUM)
WRITE (ICRK, 970) HISTO(L,10,LYNUM)
IF (HISTO(L,16,LYNUM).EQ.2 OR HISTO(L,16,LYNUM).EQ.3) WRITE (ICRK, 980) HISTO(L,10,LYNUM)
DO IX = 1, 3
  WRITE (ICRK, 990) (DCS(IX,JK), JK = 1, 3)
END DO

ELSE IF (PROPER(25,MATRL).EQ.1) THEN
  IF (LHE.1 .AND. LHE.2 .AND. LHE.3 .AND. LHE.4) GO TO 200

  WRITE (ICRK, 1030) HISTO(L,29,LYNUM)
  WRITE (ICRK, 1030) HISTO(L,30,LYNUM)
  WRITE (ICRK, 1030) HISTO(L,31,LYNUM)
  WRITE (ICRK, 1030) HISTO(L,32,LYNUM)
  WRITE (ICRK, 1030) HISTO(L,33,LYNUM)

  IF (HISTO(L,34,LYNUM).EQ.1 OR HISTO(L,34,LYNUM).EQ.2) THEN
    WRITE (ICRK, 1120) HISTO(L,34,LYNUM)
    WRITE (ICRK, 1120) HISTO(L,35,LYNUM)
    WRITE (ICRK, 1120) HISTO(L,36,LYNUM)
  ENDIF

  IF (PROPER(23,MATRL).EQ.1) THEN
    WRITE (ICRK, 1370) HISTO(L,67,LYNUM)
    WRITE (ICRK, 1380) HISTO(L,69,LYNUM)
  ENDIF

  WRITE (ICRK, 1130) HISTO(L,64,LYNUM)

  IF (HISTO(L,61,LYNUM).GE.PROPER(6,MATRL)) WRITE
    (ICRK, +)
      'UNIAXIAL STRENGTH EXCEEDED FT',
    PROPER(6,MATRL)
  WRITE (ICRK, 1080) HISTO(L,61,LYNUM)
  WRITE (ICRK, 1080) HISTO(L,62,LYNUM)
  WRITE (ICRK, 1080) HISTO(L,63,LYNUM)

  IF (HISTO(L,30,LYNUM).EQ.2 OR HISTO(L,30,LYNUM).EQ.3) THEN
    WRITE (ICRK, 1110) HISTO(L,30,LYNUM)
    WRITE (ICRK, 1110) HISTO(L,31,LYNUM)
    WRITE (ICRK, 1110) HISTO(L,32,LYNUM)
  ENDIF

  IF (HISTO(L,29,LYNUM).EQ.2 OR HISTO(L,29,LYNUM).EQ.3) THEN
    WRITE (ICRK, 1100) HISTO(L,29,LYNUM)
    WRITE (ICRK, 1100) HISTO(L,30,LYNUM)
    WRITE (ICRK, 1100) HISTO(L,31,LYNUM)
    WRITE (ICRK, 1100) HISTO(L,32,LYNUM)
  ENDIF

  DO KK = 10, 18, 3
    WRITE (ICRK, 990) HISTO(L,KK,LYNUM), HISTO(L,KK+1,LYNUM), HISTO(L,KK+2,LYNUM)
  END DO

ELSE IF (CRACK(L,1,LYNUM).GE.1.0) THEN
  WRITE (ICRK, 1090) HISTO(L,41,LYNUM)
  WRITE (ICRK, 1090) HISTO(L,42,LYNUM)
  WRITE (ICRK, 1090) HISTO(L,43,LYNUM)
  IF (HISTO(L,30,LYNUM).EQ.2 OR HISTO(L,30,LYNUM).EQ.3) THEN
    WRITE (ICRK, 1100) HISTO(L,38,LYNUM)
    WRITE (ICRK, 1100) HISTO(L,39,LYNUM)
    WRITE (ICRK, 1100) HISTO(L,40,LYNUM)
  ENDIF

  WRITE (ICRK, 1130) HISTO(L,KK,LYNUM), KK=44,46)
ENDIF
IF (CRACK(L,1,LYHUB) .GT. 0.0) THEN
    NCRACK = IST(CRACK(L,1,LYHUB))
WRITE (ICRK, 1350) NCRACK
DO 140 ICRACK = 1, NCRACK
   WRITE (ICRK, 1140) ICRACK
   WRITE (ICRK, 1150) CRACK(L,117+ICRACK,LYHUB)
   WRITE (ICRK, 1160) CRACK(L,37+ICRACK,LYHUB)
   WRITE (ICRK, 1170) CRACK(L,57+ICRACK,LYHUB)
   WRITE (ICRK, 1180) CRACK(L,67+ICRACK,LYHUB)
   WRITE (ICRK, 1190) CRACK(L,87+ICRACK,LYHUB)
   WRITE (ICRK, 1200) CRACK(L,97+ICRACK,LYHUB)
   WRITE (ICRK, 1210) CRACK(L,107+ICRACK,LYHUB)
   ICK = 9*(ICRACK - 1) + 1
   DO 1220 KKK = ICK, ICK + 8, 3
      WRITE (ICRK, 1220) CRACK(L,137+KKK,LYHUB), CRACK(L,138+KKK,LYHUB),
      CRACK(L,139+KKK,LYHUB)
   END DO
   I = 0.0
   Y1 = 0.0
   Z1 = 0.0
   DETJAC = 0.0
140 CONTINUE
DO 170 ISP = 1, 3
   WRITE (ICRK, 1230) CRACK(L,127+ICRACK,LYHUB)
   ICK = 9*(ISP - 1) + 1
   DO 1300 KKK = ICK, ICK + 8, 3
      WRITE (ICRK, 1300) TEHSTF(L,30+KKK,LYHUB), TEHSTF(L,31+KKK,LYHUB),
      TEHSTF(L,32+KKK,LYHUB)
   END DO
   IF (TEHSTF(L,21+ISP,LYHUB) . EQ. 2.0 . OR.
      TEHSTF(L,21+ISP,LYHUB) . EQ. 3.0) WRITE (ICRK, 1310) TEHSTF(L,15+ISP,LYHUB)
150 CONTINUE
170 CONTINUE
ENDIF
C* THIS PORTION IS TO OBTAIN CRACK ORIENTATION PLANE FOR POST PROCESSING
C 180 CONTINUE
IF (PROPER(25,MATL) . EQ. 1.0) THEN
   IF (ICXFGO . EQ. 1) THEN
      IF (MOD(IVCHEM,ICXTR) . EQ. 0) THEN
         IF (CRACK(L,1,LYHUB) . GE. 1.0) THEN
            X1 = 0.0
            Y1 = 0.0
            Z1 = 0.0
            DETJAC = 0.0
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SIZE = 0.0
THICKE = 0.0
RAD = 0.0
IF (IFLAG .EQ. 0) THEN
  CALL JACB3D (L, ELNUM, SHEL, LSHRDA, DETJAC)
  CALL COORD1 (ELNUM, SHEL, L, ITYPE, XI, YI, ZI)
  SIZE = (W(L)*DABS(DETJAC))**(1./3.)
ELSE IF (IFLAG .EQ. 4) THEN
  CALL GETTHK (L, ELNUM, SHEL, THICKE, RAD)
  C VECTOR V3 RETURNED BY JACSEL IS NORMAL TO MID SURFACE OF THE
  SHELL AT INTEGRATION POINTS
  ISET = 0
  SIP = SI(L)
  CALL JACSEL (L, ELNUM, SHEL, TRICKE, DETJAC, V3, ISET, SIP)
  CALL COORD1 (ELNUM, SHEL, L, ITYPE, XI, YI, ZI)
  SIZE = (W(L)*DABS(DETJAC))**(1./3.)
ENDIF

BCrack = I1T(Crack(L,1,LYNUM))
DO KP = 1, BCrack
  KX = 9*(KP-1) + 1
  AL1 = Crack(L,137+KX,LYNUM)
  AM1 = Crack(L,138+KX,LYNUM)
  AL2 = Crack(L,140+KX,LYNUM)
  AM2 = Crack(L,141+KX,LYNUM)
  AL3 = Crack(L,143+KX,LYNUM)
  AM3 = Crack(L,144+KX,LYNUM)
  AB2 = Crack(L,146+KX,LYNUM)
  ISTAT = 1
  IF (Crack(L,117+KP,LYNUM) .EQ. 1) THEN
    ISTAT = 1
  ELSE IF (Crack(L,117+KP,LYNUM) .EQ. 2) THEN
    ISTAT = 2
  ELSE IF (Crack(L,117+KP,LYNUM) .EQ. 3) THEN
    ISTAT = 2
  EI
  XI = XI + AL2*SIZE/2.0
  YA = YI + AM2*SIZE/2.0
  ZA = ZI + AR2*SIZE/2.0
  XB = XI + (AL2*SIZE/2.0)
  YB = YI + (AM2*SIZE/2.0)
  ZB = ZI + (AR2*SIZE/2.0)
  XC = XI + AL3*SIZE/2.0
  YC = YI + AM3*SIZE/2.0
  ZC = ZI + AR3*SIZE/2.0
  XD = XI + (AL3*SIZE/2.0)
  YE = YI + (AM3*SIZE/2.0)
  ZD = ZI + (AR3*SIZE/2.0)
WRITE (ICK2, 1390) IILISE, ISTAT
WRITE (ICK2, 1400)
WRITE (ICK2, 1410)
WRITE (ICK2, 1420) XA, YA, ZA
WRITE (ICK2, 1430) ILHE + 1, ISTAT
WRITE (ICK2, 1400)
WRITE (ICK2, 1410)
WRITE (ICK2, 1420) XC, YC, ZC
WRITE (ICK2, 1430) ILHE + 2, ISTAT
WRITE (ICK2, 1400)
WRITE (ICK2, 1410)
WRITE (ICK2, 1420) XB, YB, ZB
WRITE (ICK2, 1430) ILHE + 3, ISTAT
WRITE (ICK2, 1400)
WRITE (ICK2, 1410)
WRITE (ICK2, 1420) XD, YD, ZD
WRITE (ICK2, 1430) ILHE = ILHE + 4
END DO
ENDIF
ENDIF
END

C
WRITE(*,*) 'ICKFGO', ICKFGO, 'ICCREM', INCREM
IF (MOD(INCREM,ICCREM) .EQ. 0) WRITE (ICK2, '(AE)') ' -1'
ENDIF
RETURN
900 FORMAT(IOX,D12.5, 'IS THE STRESS POINT STATUS')
910 FORMAT(IOX,D12.5, 'IS THE LOADING STATUS')
920 FORMAT(IOX,D12.5, 'IS THE CURRENT STIFFNESS')
930 FORMAT(IOX,D12.5, 'IS THE CURRENT STRAIN')
940 FORMAT(IOX,D12.5, 'IS THE CURRENT STRESS')
950 FORMAT(IOX,D12.5, 'IS THE INITIAL STIFFNESS')
960 FORMAT(IOX,D12.5, 'IS THE STRAIN AT YIELD')
970 FORMAT(IOX,D12.5, 'IS THE STRESS AT YIELD')
980 FORMAT(IOX,D12.5, 'IS THE STRAIN AT UN/RELOAD')
990 FORMAT(IOX,3(D12.5, 'ARE THE DIR. COSINES')
1000 FORMAT(IOX,IS, 'ARE THE ELEMENT NO.', '************')
1010 FORMAT(IOX,IS, 'ARE THE STEEL LAYER NO.', '************')
1020 FORMAT(IOX,IS, 'ARE THE CONCRETE LAYER NO.', '************')
1030 FORMAT(IOX, '************INTACT CONCRETE INFORMATION', '************')
1040 FORMAT(IOX,D12.5, 'IS THE CURRENT SECANT POISSONS RATIO')
1050 FORMAT(IOX,D12.6, 'IS THE CURRENT STRESS STATE REGION')
1060 FORMAT(IOX,D12.6, 'IS THE FAILURE FUNCTION VALUE')
1070 FORMAT(IOX,D12.6, 'IS THE PRINCIPAL STRAIN INTACT CONCRETE VALUE')
1080 FORMAT(IOX,D12.6, 'IS THE PRINCIPAL STRAIN VALUE')
1090 FORMAT(IOX,D12.6, 'IS THE TOTAL PRINCIPAL STRAIN VALUE')
1100 FORMAT(IOX,D12.6, 'TOTAL PRINCIPAL STRAIN UN/RELOAD VALUE')
1110 FORMAT(IOX,D12.6, 'PRINCIPAL STRESS VALUE AT FAILURE')
1120 FORMAT(IOX,D12.6, 'PRINCIPAL STRAIN UN/RELOAD VALUE')
1130 FORMAT(IOX,3(D12.5, 'DIR. COSINE CRACKED CONCRETE')
1140 FORMAT(IOX,IS, 'CURRENT CRACK NO.')
1150 FORMAT(IOX,D12.6, 'IS THE STATE OF THE CRACK')
1160 FORMAT(IOX,D12.6, 'IS THE CRACK INITIATION STRESS')
1170 FORMAT(IOX,D12.6, 'CRACK NORMAL STRAIN CURRENT')
1180 FORMAT(10X,D12.6,'IS THE CRACK SHEAR STRAIN CURRENT')
1190 FORMAT(10X,D12.6,'IS THE CRACK NORMAL STIFFNESS')
1200 FORMAT(10X,D12.6,'IS THE CRACK SHEAR STIFFNESS')
1210 FORMAT(10X,D12.6,'IS THE CRACK TERM. STRAIN')
1220 FORMAT(10X,3(D12.8),'ARE DIR. COSINE OF CRACK')
1230 FORMAT(10X,D12.6,'IS THE CRACK UNLOADING STRAIN')
1240 FORMAT(10X,D12.6,'IS THE STATE OF TEB. STIF. SPR.')
1250 FORMAT(10X,D12.6,'IS THE INITIATION STRESS')
1260 FORMAT(10X,D12.6,'IS THE CURRENT STRAIN')
1270 FORMAT(10X,D12.6,'IS THE CURRENT STIFFNESS')
1280 FORMAT(10X,D12.6,'IS THE STRAIN AT INITIATION')
1290 FORMAT(10X,D12.6,'IS THE STRAIN AT TERMINATION')
1300 FORMAT(10X,3(D12.8),'ARE DIR. COSINE OF SPRING')
1310 FORMAT(10X,D12.6,'IS THE STRAIN AT US/RELOADING')
1320 FORMAT(1X,16,'*****IS THE INTEGRATION POINT #','**************',/)
1330 FORMAT(1X,16,'INCREMENT NO. ','**')
1340 FORMAT(1X,16,'ITERATION NO. ','**')
1350 FORMAT(1X,16,'*CRACK INFORMATION---NO. OF CRACKS','**',/)
1360 FORMAT(1X,16,'*STRESS STIFFENING SPRING NO.','**',/)
1370 FORMAT(10X,D12.6,'IS THE SECANT STIFFNESS AT FAILURE')
1380 FORMAT(10X,D12.6,'IS THE POISSONS RATIO AT FAILURE')
1390 FORMAT(10X,9X,'0','9X','0','9X','0','9X','0','9X','0','9X','0')
1400 FORMAT(9X,'0','9X','0','9X','0','9X','0','9X','0')
1410 FORMAT(9X,'1','9X','1')
1420 FORMAT(1P,3D26.16)
1430 FORMAT(1X)
1440 FORMAT(180))

END

C*******************************************************************************
C SUBROUTINE OUTPUT(VALuetIOUT,ERROR)
C IMPLICIT REAL*8(A-H,O-Z)
C . . .SWITCHES: RENUM=100:10,FORMAT=900:10
C . . .SWITCHES:
C*******************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C OUT1 OUT2 OUT3 OUT4 OUT7 OUT10
C OUT8 OUT9 OUT20 OUT20 OUT21 OUT22
C*******************************************************************************

REAL*8 VALUE( *)
COMMON/OUTPT1/IDFLAG,IFCAED
C
C ---- DESCRIPTION OF THE OUTPUT MODULES AND THE CORRESPONDING BIT
C FLAG POSITION IS THE IDFLAG INTEGER VARIABLE
C
C FILE BIT DESCRIPTION
C
C OUT1 1 OUTPUT ELEMENT STRESSES AT THE INTEGRATION POINTS
C OUT2 2 OUTPUT ELEMENT STRESSES AT THE INTEGRATION POINTS
C OUT3 3 OUTPUT ELEMENT STRESSES AT THE ELEMENT NODES
C OUT4 4 OUTPUT ELEMENT STRAINS AT THE ELEMENT NODES
C OUT5 5 OUTPUT AVERAGE STRESSES AT THE NODES
C OUT6 6 OUTPUT AVERAGE STRESSES AT THE NODES
C OUT7 7 OUTPUT DISPLACEMENTS AT THE NODES
C OUT8 8 OUTPUT RODAL EQUILIBRIUM FORCES AT THE NODES
C OUT9 9 OUTPUT RODAL EQUILIBRIUM FORCES FOR FREEBODIES
C OUT10 10 OUTPUT REACTIONS AT THE SUPPORTS
C OUT11 11 OUTPUT RODAL COORDINATES
C OUT12 12 OUTPUT ELEMENT CONNECTIVITY
THE FOLLOWING ARE CASES UNIVERSAL DATASETS

---

OUT20 20 OUTPUT SECOND PIOLA KIRCHHOFF STRESSES AT THE ELEMENT NODES
OUT21 21 OUTPUT CAUCHY STRESSES AT THE ELEMENT NODES
OUT22 22 OUTPUT TOTAL ELEMENT STRAINS AT THE ELEMENT NODES
OUT23 23 OUTPUT ELEMENT ELASTIC STRAINS AT THE ELEMENT NODE
OUT24 24 OUTPUT PLASTIC ELEMENT STRAINS AT THE ELEMENT NODE
OUT25 25 OUTPUT NODAL DISPLACEMENTS
OUT26 26 OUTPUT SUPPORT REACTIONS
OUT27 27 OUTPUT NODAL EQUILIBRIUM LOADS

C

IF (BTEST(10FLAG,1)) CALL OUT1 (IOUT)
IF (BTEST(10FLAG,2)) CALL OUT2 (IOUT)
IF (BTEST(10FLAG,3)) CALL OUT3 (IOUT)
IF (BTEST(10FLAG,4)) CALL OUT4 (IOUT)
IF (BTEST(10FLAG,5)) CALL OUT5 (IOUT)
IF (BTEST(10FLAG,6)) CALL OUT6 (IOUT)
IF (BTEST(10FLAG,7)) CALL OUT7 (IOUT)
IF (BTEST(10FLAG,8)) CALL OUT8 (IOUT)
IF (BTEST(10FLAG,9)) CALL OUT9 (IOUT)
IF (BTEST(10FLAG,10)) CALL OUT10 (IOUT)
IF (BTEST(10FLAG,11)) CALL OUT11 (IOUT)

C

IFCAED = 1
IF (1 .EQ. 1) THEN
IF (BTEST(10FLAG,20)) CALL OUT20 (IOUT, 1)
IF (BTEST(10FLAG,22)) CALL OUT20 (IOUT, 3)
IF (BTEST(10FLAG,26)) CALL OUT21 (IOUT)
IF (BTEST(10FLAG,25)) CALL OUT22 (IOUT)
ENDIF

RETURN
END

C

C  =========================== OUT1 =====================
C
C  INCLUDE (PROCESS)
SUBROUTINE OUT1(IOUT)
IMPLICIT REAL'S (A 'H,0-Z)

C  . . .SWITCHES: RENUMB=100:10,FRACT=900:10
C  . . .SWITCHES:
C
SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO ELINTM LYINFO ISH3DG ISHHEL C2ORD1
C IOGET REXP
C
C******************************************************************************
REAL*4 THICK
REAL*8 LTHICK
CHARACTER*67 CTEMP
INTEGER ELNUM
COMMON/UTIL1/STRESS(6),STRAIN(6),STREA(6),CTEMP
COMMON/INPUT1/BIPI3,RIPI4,RIPI5,BIP6,INTER
COMMON/INPUT2/RENOE,RENOE,BIRD,BIRD,MTIP,IFLAG1,IFLAG2,IDIM,
1 NISODE,NCOLOR,NFREE
COMMON/ICR1/FRAC(10),ELINC(10),LCQGET,LCQPTR
COMMON/INPUT2/MATYPE(10)
COMMON/DEVICE/LEDEV1,LEDEV2,LEDEV3,LEDEV4,LDDEV1,LDDEV2,LEDEV5
COMMON/INPUT3/THICK(9),IFLAG
COMMON/LAYERS/LTHICK(9),BIS(9),DCS(3,3),MLAYS,MLATL,LYNUM
COMMON/OUTPT2/IDEL(6000),IDBDD(10000)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
DIMENSION COORDS(3),CSTR(6)

IEWE = 0
DO 130 ELMUM = 1, ELEEM
  CALL ELINFO (ELUM, ITYPE, NHUE, IFLAG, ISTART, LINES)
  CALL ELITM (ELUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
              MATNUM, THICK)
  IF (IDEL(ELUM) .EQ. 1) WRITE (IDOUT, 910) ELMUM
  CALL ELIHFO (ELUM, ITYPE, HHEL, IFLAG, ISTART, LIHELS)
  CALL ELIHTH (ELUM, IDEHT, IHTCOD, HIPXI, HIPETA, HIPSI, THICK)
  IF (10EL(ELUM) .EQ. 1) WRITE (LOUT, 910) ELUM

DO 110 LNUM = 1, LAYERS
  IF (10EL(LNUM) .EQ. 1) WRITE (LOUT, 920) LNUM
  CALL LYINFO (LNUM, LTICK, ZS, HATRL, DCS, NHUE, LNUM, HIPXI, HIPETA, HIPSI)

IF (ITYPE .LE. 0) GO TO 120
IF (ITYPE .GT. 300) THEN
  IF (IFLAG .EQ. 1) THEN
    CALL ISH3DG (ITYPE, HHEL, TERROR)
  ELSE IF (IFLAG .EQ. 4) THEN
    CALL ISH5HL (ITYPE, HHEL, TERROR)
  ENDIF
  IEWE = 6
ENDIF
  IF (ITYPE1 = ITYPE) THEN
    IF (10EL(ELUM) .EQ. 1) WRITE (LOUT, IF2)
    DO 100 ITOPS = 1, EIP
      CALL COORD1 (ELUM, HHEL, ITOPS, ITYPE, COORDS(1),
                   COORDS(2),COORDS(3))
    CALL IODET (LDEV1, 96, ' (A96)', 96)
    IF (10EL(ELUM) .EQ. 1) WRITE (LOUT, IFOR) ITOPS, (COORDS(KI), KI = 1, IDIM), (STRESS(KI), KI = 1, IEHD)
  ELSE IF (ITYPE .EQ. 0 OR IDETH .EQ. 0) THEN
    CALL COORD1 (ELUM, HHEL, ITOPS, ITYPE, COORDS(1),
                   COORDS(2),COORDS(3))
    CALL IDDET (LDEV1, 96, ' (A96)', 96)
    IF (10EL(ELUM) .EQ. 1) WRITE (LOUT, IFOR) ITOPS, (COORDS(KI), KI = 1, IDIM), (STRESS(KI), KI = 1, IEHD)
    CONTINUE
  ELSE
    RETURN
  ENDIF

RETURN

900 FORMAT(16,1P,6E14.6)
910 FORMAT(/,20X,'S T R E S S E S A T I N T E G R A T I O N ',
         1 'P O I N T S O N E L E M E N T ',IT)
920 FORMAT(/,20X,'S T R E S S E S A T I N T E G R A T I O N ',
         1 'P O I N T S O F L A Y E R ',IT)
930 FORMAT(34X,1P,4E14.6)
940 FORMAT(1X,'POIN',13X,'X',13X,'Y',
         1 11X,'SIX',11X,'SY',11X,'SZ')
C = INCLUDE (PROCESS)

SUBROUTINE OUTF2 (IOUT)
IMPLICIT REAL*8 (A-H,O-Z)

C . . SWITCHES: REBUBB=100:10, FDRHAT=900:10
C . . SWITCHES:

C SUBROUTINES ADD FUCCTIONS CALLED FROM THIS ROUTINE
C
C ELBFO ELITM LYTIFG ISHGDI ISHSHL IOGT
C CODRD1REW
C
C**************************************************************

REAL*4 THICK
REAL*8 LTHICK
CHARACTER*57 CTEMP
INTEGER ELBUB
COMMOD/LAYERS/LTHICK(9),LS(9),DCS(3,3),LAYRS,MATL,LYBUB
COMMOD/UTILI/STRESS(6),STRAIN(6),STRELA(6),CTEMP
COMMOD/IPUTI/EBPSL,EBPSL,EBPSL,EBPSL,EBPSL,EBPSL
COMMOD/IPUT6/BRUNES,EBLEM,BRDF,BLINC,MBIT,IFLAG1,IFLAG2,IDIM,
1 NODE,NCOLR,NFREE
COMMOD/EBUM1/FRAC(10),EBUM1(10),LOGM1,IMCPTR
COMMOD/IPUT1/MATYPE(10)
COMMOD/IPUT2/TREE(9),IFLAG
COMMOD/DEVICE/LEV1,LEV2,LEV3,LEV4,LEV5,LEV6,LDKPEP,LEV6,LEDSST
COMMOD/OUPT2/IDEL(5000),IDNB(10000)
DIMENSION COORD(9),STRELA(6)

IEED = 0
DO 130 ELBUM = 1, BELEM
   CALL ELBFO (ELBUM, ITYPE, NLUM, IFLAG, ISTAT, LINES)
CALL ELIBM (ELBUM, IDEBT, ISTCOD, NIPEI, NIPETA, NIPESI, MATNUM, THICK)
C
IF (IDEL(ELBUM) .EQ. 1) WRITE (IOUT, 910) ELBUM
DO 11 IYBUM = 1, NLAYAS
IF (IDEL(ELBUM) .EQ. 1) WRITE (IOUT, 920) IYBUM
CALL LYIBFO (LYBUM, LSTICK, ZS, MATEL, DCS, IBEEL, ELBUM, 
       NIPEI, NIPETA, NIPESI)
C
IF (ITYPE .LE. 0) GO TO 120
IF (ITYPE .GT. 0 .OR. IDEBT .eq. 0) THEN
  IF (ITYPE .GT. 300) THEN
    CALL 990 TO IFOR
    ASSIGN 990 TO IFOR1
    IF (MATYPE(MATBUM) .EQ. 2) THEN
      ASSIGN 1010 TO IF1
      ELSE
      ASSIGN 1000 TO IF1
      ENDIF
      IF (IFLAG .EQ. 0) THEN
        CALL ISH3DG (ITYPE, BBEL, TERROR)
      ELSE IF (IFLAG .EQ. 4) THEN
        CALL ISHSHL (ITYPE, BBEL, TERROR)
      ENDIF
      IENDD = 6
      ENDIF
    IF (ITYPE1 = ITYPE)
    IF (IDENT1 = IDEBT)
      IF (IDEL(ELBUM) .EQ. 1) WRITE (IOUT, IF1)
      DO 100 ISTOP = 1, NIP
        CALL IDGET (LEVI, 96, 'dV96'), S
        CALL COORD1 (ELBUM, BBEL, ISTOP, ITYPE, COORDS(1), 
          1, COORDS(2), COORDS(3))
        IF (IDEL(ELBUM) .EQ. 1) WRITE (IOUT, IFOR) ISTOP, 
          1, COORDS(K1), K1 = 1, IDIM), (STRAIN(K1), K1 = 1, 
          2, IENDD)
      CONTINUE
    ELSE
      CALL REWIE
      RETURN
    C
    990 FORMAT(I6,1P,6E14.6)
    910 FORMAT(/,20X,'ST Ra I N S AT I NTEGRATION ', 
          1, 'PO I NTS ON ELEMENT', I7)
    920 FORMAT(/,20X,'ST Ra I N S AT I NTEGRATION ', 
          1, 'PO I NTS ON L A Y E R', I7)
    C
    930 FORMAT(2(34X,1P,4E14.6/))
    C
    940 FORMAT(1X,'POINT',13X,'X',13X,'Y', 
          11X,'EXX',11X,'EYY',11X,'EYX',11X,'EZZ')
    C
    950 FORMAT(1X,'POINT',13X,'X',13X,'Y', 
          16X,'TOTAL_X' '3X,' TOTAL_Y' '3X,' TOTAL_XY' '2X, 
          2' TOTAL_Z' '2' 
          340X,'ELAST_X',6X,'ELAST_Y',6X,'ELAST_XY',6X,'ELAST_Z' 
          440X,'PLAST_X',6X,'PLAST_Y',6X,'PLAST_XY',6X,'PLAST_Z' /)
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
C 980 FORMAT(3E12.1P, 9E14.B)
C 990 FORMAT(2(48X, 1P, 6E14.B))
C
C END
C
C ***************************************** 0 U T 3 *****************************************
C
C SUBROUTINE DUT3COUT
C IMPLICIT REAL*8 (A-H, O-Z)
C
C SWITCHES: RENUMB = 100: 10, FORMAT = 900: 10
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO, ELINTH, LAYRO, IGET, REWIS
C
C REAL*4 THICK
C REAL*8 LTHICK
C CHARACTER*57 CTMPL
C INTEGER ELNUM
C COMMON/XEB/LTHICK(9), DCPS(3, 3), ELPLST, MATEL, LYNUM
C COMMON/UTPST/STRESS(6), STRAIN(6), STELA(6), CTMPL
C COMMON/IPUTS/BISPT, BISPTA, BISPTI, INTPOD
C COMMON/IPUTS2/SOP(20), 6000)
C COMMON/IPUTS/BMODES, BELEM, MDF, ELBEC, MTT, IFLAG1, IFLAG2, IDIM,
C 1 EISCODE, NCOLE, NFREE
C COMMON/IBCID11/FRACT(10), ELINC(10), LDCST, ISCPTR
C COMMON/IPUTP/MATYPE(10)
C COMMON/ISPCD/F(64, 27)
C COMMON/DEVICES/LDEV1, LDEV2, LDEV3, LDEV4, LDEV5, LDEVS, LDEVN, LDEVST
C COMMON/INPUT9/TEICK(9), JFLAG
C COMMON/OUTPUT2/10DE(6000), LQDOP(10000)
C DIMENSION CAUC(6, 27), CAUCH(6), STUS(6, 27)
C
C IEND = 0
D0 180 ELNUM = 1, BELEM
CALL ELINFO (ELNUM, ITYPE, BELEM, IFLAG, ISTAT, LINES)
CALL ELSTM (ELNUM, IDENT, IBTCD, HIPX, HIPETA, HIPSI, MATNUM, THICK)

DO 160 LYBUM = 1, BLAYRS
   CALL LYIBFO (LYBUM, LTHICK, ZS, MATRL, DCS, NEEL, ELNUM, HIPX, HIPETA, HIPSI)
   IF (ITYPE .LE. 0) GO TO 170
   IF (ITYPE .NE. 0 .OR. IDENT .NE. 0) THEN
      CALL LAGRGl (ITYPE, IFLAG, NEEL, IERROR)
      IF (ITYPE .GT. 300) THEN
         ASSIGN 980 TO IFOR
         ASSIGN 990 TO IFOR1
         ASSIGN 1000 TO IF2
         IEHD = 6
      ENDIF
   ENDIF
   DO 110 K1 = 1, NEEL
      DO 100 K2 = 1, IEHD
         CAUC(K2,K1) = 0.
         STRS(K2,K1) = 0.
      CONTINUE
   DO 110

   DO 140 INTYPE = 1, HIP
      CALL IZGET (LDEY1, 96, '(A96)', 5)
      DO 130 K1 = 1, NEEL
         DO 120 K2 = 1, IEHD
            STRS(K2,K1) = STRS(K2,K1) + STRRESS(K2)*P(1,INTYPE,K1)
         CONTINUE
      DO 130
   DO 140

   IF (IOEL(ELNUM) .EQ. 1) WRITE (IOOUT, 910) ELNUM
   IF (IOEL(ELNUM) .EQ. 1) WRITE (IOOUT, 920) LYBUM
   IF (IOEL(ELNUM) .EQ. 1) WRITE (IOOUT, IF2)
   DO 180 K1 = 1, NEEL
      IF (IOEL(ELNUM) .EQ. 1) WRITE (IOOUT, IFOR) K1, NOP(K1, ELNUM), (STRS(I,K1), I = 1, IEHD)
   CONTINUE

RETURN

FORMAT(6,I10,1P,4E14.6)
910 FORMAT(20X,'STRESSES AT THE NODES ', 1 'ON ELEMENT',17)
920 FORMAT(20X,'STRESSES OF THE LAYER #', 1 'AT NODE LEVEL',17)
930 FORMAT(16X,1P,4E14.6)
940 FORMAT(I9,'NODE',3X,'NODE ID.', 1 11X,'SXX',11X,'SYY',11X,'SXY',11X,'SZZ')
950 FORMAT(I9,'NODE',3X,'NODE ID.'),
SUBROUTINE 0UT4 (IOUT)

IMPLICIT REAL*8 (A-H,O-Z)

C ...SWITCHES: RENUM=100:10,FORMAT=900:10
C ...SWITCHES:

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO ELIBTM LYINFO LAG6G1 IOGET REWIN
C
C=======================================================================

REAL*8 LTHICK
REAL*4 THICK
CHARACTER*57 CTEMP

INTEGER ELMU

COMMON/LAYER/LTEICK(9),2S(9),DCS(3,3),MLVRS,MATRL,LYNUM
COMMON/UTIL1/STRESS(6),STRAIN(6),STRELA(6),CTEMP
COMMON/INPUT2/BIPZI,ZIPETA,BIPSI,BIP,ISTCOD
COMMON/INPUT4/NODES,NELEM,NDOP,NDISP,BMAT,IFLAG1,IFLAG2,IDIM,
  NNODE,NCOLOR,MPASS
COMMON/IBR11/FRAC(10),BLISC1(10),LDCOST,ICODPR
COMMON/INPUT2/IDOP(20,5000)
COMMON/INPUT1/MASTPS(10)
COMMON/INPUTS/MASTPS(10)
COMMON/ISPCONS/P(64,27)
COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDEV8,LDEVS,LDEVE,LDEVST
COMMON/INPUT2/THICK(9),IFLAG
COMMON/OUTPT2/IDEL(5000),IOBSQ(10000)

DIMENSION STRM(8,20),ELSTM(8,20),STRPLA(6)

COMMON/IOUP/0

DO 180 ELMU = 1, SELEM
  CALL ELINFO (ELNUM, ITYPE, MNEL, IFLAG, ISTART, RINES)
  CALL ELBM (ELNUM, IDENT, ISTD, REWIN, RINE,
          ZIP, ZIPETA, RIPS1, RIPS2, RIPS3, RIPS4,
          RIPS5, RIPS6, RIPS7, RIPS8, RIPS9, RIPS10,
          RIPS11, RIPS12, RIPS13, RIPS14, RIPS15,
          RIPS16, RIPS17, RIPS18, RIPS19, RIPS20,
          RIPS21, RIPS22, RIPS23, RIPS24, RIPS25,
          RIPS26, RIPS27, RIPS28, RIPS29, RIPS30,
          RIPS31, RIPS32, RIPS33, RIPS34, RIPS35,
          RIPS36, RIPS37, RIPS38, RIPS39, RIPS40,
          RIPS41, RIPS42, RIPS43, RIPS44, RIPS45,
          RIPS46, RIPS47, RIPS48, RIPS49, RIPS50,
          RIPS51, RIPS52, RIPS53, RIPS54, RIPS55,
          RIPS56, RIPS57, RIPS58, RIPS59, RIPS60,
          RIPS61, RIPS62, RIPS63, RIPS64, RIPS65,
          RIPS66, RIPS67, RIPS68, RIPS69, RIPS70,
          RIPS71, RIPS72, RIPS73, RIPS74, RIPS75,
          RIPS76, RIPS77, RIPS78, RIPS79, RIPS80,
          RIPS81, RIPS82, RIPS83, RIPS84, RIPS85,
          RIPS86, RIPS87, RIPS88, RIPS89, RIPS90,
          RIPS91, RIPS92, RIPS93, RIPS94, RIPS95,
          RIPS96, RIPS97, RIPS98, RIPS99, RIPS100,
          RIPS101, RIPS102, RIPS103, RIPS104,
          RIPS105, RIPS106, RIPS107, RIPS108,
          RIPS109, RIPS110, RIPS111, RIPS112,
          RIPS113, RIPS114, RIPS115, RIPS116,
          RIPS117, RIPS118, RIPS119, RIPS120,
          RIPS121, RIPS122, RIPS123, RIPS124,
          RIPS125, RIPS126, RIPS127, RIPS128,
          RIPS129, RIPS130, RIPS131, RIPS132,
          RIPS133, RIPS134, RIPS135, RIPS136,
          RIPS137, RIPS138, RIPS139, RIPS140,
          RIPS141, RIPS142, RIPS143, RIPS144,
          RIPS145, RIPS146, RIPS147, RIPS148,
          RIPS149, RIPS150, RIPS151, RIPS152,
          RIPS153, RIPS154, RIPS155, RIPS156,
          RIPS157, RIPS158, RIPS159, RIPS160,
          RIPS161, RIPS162, RIPS163, RIPS164,
          RIPS165, RIPS166, RIPS167, RIPS168,
          RIPS169, RIPS170, RIPS171, RIPS172,
          RIPS173, RIPS174, RIPS175, RIPS176,
          RIPS177, RIPS178, RIPS179, RIPS180,
          RIPS181, RIPS182, RIPS183, RIPS184,
          RIPS185, RIPS186, RIPS187, RIPS188,
          RIPS189, RIPS190, RIPS191, RIPS192,
          RIPS193, RIPS194, RIPS195, RIPS196,
          RIPS197, RIPS198, RIPS199, RIPS200,
          RIPS201, RIPS202, RIPS203, RIPS204,
          RIPS205, RIPS206, RIPS207, RIPS208,
          RIPS209, RIPS210, RIPS211, RIPS212,
          RIPS213, RIPS214, RIPS215, RIPS216,
          RIPS217, RIPS218, RIPS219, RIPS220,
1 MATSUM, THICK

DO 160 LYNUM = 1, HLAYRS
   CALL LYNFSO (LYNUM, LTHICK, ZS, MATAL, DCS, NBEL, ELNUM, 
   HIFX1, HIFST1, HIEF1)
C
   IF (ITYPE .LE. 0) GO TO 170
   IF (ITYPE .EQ. 0 .OR. IDENT .NE. 0) THEN
      CALL LAGRGl (ITYPE, IFLAG, NBEL, IERROR)
      IF (ITYPE .GT. 300) THEN
         ASSIGN 980 TO IFOR
         ASSIGN 990 TO IFORI
         ASSIGN 1000 TO IF2
         IEND = 0
      ENDIF
   ENDIF
C   ITYPE1 = ITYPE
C   IDENT1 = IDENT
C
   DO 110 K1 = 1, NBEL
      DO 100 K2 = 1, IEND
         STRH(K2,K1) = 0.
         ElSTR(K2,K1) = 0.
      100 CONTINUE
   110 CONTINUE
C
   DO 140 ISTOPS = 1, IIP
      CALL IISSET (LDEV1, 96, '(A96)', 5)
C
   DO 130 K1 = 1, NBEL
      NODE = EOP(K1,ELNUM)
      DO 120 K2 = 1, IEND
         STRH(K2,K1) = STRH(K2,K1) + STRAIH(K2)*P( 
            ISTOPS,K1)
      120 CONTINUE
   130 CONTINUE
   140 CONTINUE
C
   IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, 910) ELNUM
   IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, 920) LYNUM
   IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, IF2)
   DO 150 K1 = 1, NBEL
      IF (IOEL(ELNUM) .EQ. 1) WRITE (IOUT, IFOR) K1, EOP(K1, 
         ELNUM), (STRH(I,K1), I = 1, IEND)
   150 CONTINUE
C   160 CONTINUE
C   170 CONTINUE
   CALL REWIS
C
   RETURN
C
900 FORMAT(16,I10,1P,4E14.5)
910 FORMAT(17,'STRAINS AT THE NODES ', 1 'O H E L E M E N T',17)
920 FORMAT(17,'STRAINS AT LAYE RS ', 1 'AT THE NODE LEVEL',17)
930 FORMAT(2(16I,1P,4E14.5/))
C
940 FORMAT(11I,'NODE',3X,'NODE ID.',1 11I,'EXE',11X,'EYY',11X,'EXY',11X,'EXZ')
C
950 FORMAT(1X,'NODE',3X,'NODE ID. ',
1 3X, 'TOTAL_X',3X,'TOTAL_Y',2X,'TOTAL_Z',3X,
2 'TOTAL_S'/
3 21X,'ELASTIC_X',5X,'ELASTIC_Y',4X,'ELASTIC_Z',5X,
4 'ELASTIC_T'/
5 21X,'PLASTIC_X',5X,'PLASTIC_Y',4X,'PLASTIC_Z',5X,
6 'PLASTIC_T'/)
C
960 FORMAT(1X,'NODE',3X,'NODE ID. ',
1 11X,'ER ',11X,'EY ',11X,'ER_Y ',11X,'ET'/)
C
970 FORMAT(1X,'NODE',3X,'NODE ID. ',
1 3X, 'TOTAL_R',3X,'TOTAL_Y',2X,'TOTAL_Z',3X,
2 'TOTAL_T'/
3 21X,'ELASTIC_R',5X,'ELASTIC_Y',4X,'ELASTIC_Z',5X,
4 'ELASTIC_T'/
5 21X,'PLASTIC_R',5X,'PLASTIC_Y',4X,'PLASTIC_Z',5X,
6 'PLASTIC_T'/)
C
980 FORMAT(IS,II1,IP,9EI4.5)
990 FORMAT(2EI16X,1P,6EI4.5/))
C
1000 FORMAT(1X,'NODE',3X,'NODE ID. ',
1 11X,'EXX',11X,'EYY',11X,'EZZ',11X,'EXY',11X,'EYZ',11X,'EXZ'/)
C
1010 FORMAT(1X,'NODE',3X,'NODE ID. ',
1 16X, 'TOTAL_X',3X,'TOTAL_Y',3X,'TOTAL_Z',3X,
2 'TOTAL_XY',2X,'TOTAL_XZ',2X,'TOTAL_YZ',2X,
3 22X,'ELAST_X',6X,'ELAST_Y',6X,'ELAST_Z',6X,'ELAST_XY',6X,
4 'ELAST_YZ',6X,'ELAST_XZ'/
5 22X,'PLAST_X',6X,'PLAST_Y',6X,'PLAST_Z',6X,'PLAST_XY',6X,
6 'PLAST_YZ',6X,'PLAST_XZ'/)
C
END

C ******************************************************************************
C INCLUDE (PROCESS)
C SUBROUTE OUT7(IOUT)
C IMPLICIT REAL*8 (A-H,O-Z)
C SWITCHES: RESUMB=100:10,FORMAT=900:10
C SWITCHES:
C******************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C DIRVEC
C
C******************************************************************************
C REAL*4 SHELLZ
C COMMON/INPUTS/ENDS,WELEM,MBDF,BLINC,BSET,IFLAG1,IFLAG2,IDIM,
1 BIMODE,ICON,BFREE
C COMMON/IRCR1/FRACT(10),BLINC1(10),LDCOST,IRCPT
C COMMON/ISPUTE/ISPB(10000)
C COMMON/DEVICE/IDEV1,IDEV2,IDEV3,IDEV4,LDKEEP,LDEV,LDEVST
C COMMON/MATRIX/UTOTAL(60000)
C COMMON/TRANS/DC(3,3)
C COMMON/SHL2/SHEL2(3,10000)
C COMMON/OUTPT2/IDIM(8000),IDMOD(10000)
C DIMENSION DUMMY(6)
C

IF (IDIM .EQ. 3) THEN
  DO 110 K1 = 1, NSIDES
    I = NDIF*(K1-1)
    ICODE = IABD(ISPB(K1),256)
    ISPs = IABD(ISPB(K1),4)
    I = I + IDIM
    IF (ICODE.GT.0 .AND. ISPs.NE.4) THEN
      DC(1,3) = DBLE(SHELLZ(1,K1))
      DC(2,3) = DBLE(SHELLZ(2,K1))
      DC(3,3) = DBLE(SHELLZ(3,K1))
      CALL DIRVEC (DC(1,1),DC(1,2),DC(1,3))
      DO 100 K2 = 1, 3
        CST = 0.
        CST1 = 0.
        IDIR = I + 1
        CST1 = CST1 + UTOTAL(IDIR)*DC(K2,1)
        IDIR = I + 2
        CST1 = CST1 + UTOTAL(IDIR)*DC(K2,2)
        IDIR = I + 3
        CST1 = CST1 + UTOTAL(IDIR)*DC(K2,3)
        DUMMY1(K2) = CST1
      CONTINUE
    100 CONTINUE
  ELSE
    IDIR = I + 1
    UTOTAL(IDIR) = DUMMY1(1)
    IDIR = I + 2
    UTOTAL(IDIR) = DUMMY1(2)
    IDIR = I + 3
    UTOTAL(IDIR) = DUMMY1(3)
  ENDIF
  110 CONTINUE
ENDIF

C
WRITE (IOUT, 910)
DO 120 K1 = 1, NSIDES
  IF (BTEST(ISPB(K1),10)) THEN
    ID = NSDF*(K1-1)
    IF (ID.NE.NSDF) THEN
      IF (ID .EQ. 1) WRITE (IOUT, 900) K1, (UTOTAL(ID)+K2)
    ENDIF
  ENDIF
120 CONTINUE
C
RETURN

900 FORMAT(I11,1P,6E13.6)
910 FORMAT(/1H1,20X,'W O D A L T R A N S L A T I O N S A N D ',
      1 ' R O T A T I O N S /1X,' , ' W O D E N O .',
      2 '1X,' 'UX', '11X,' 'UY', '11X,' 'UZ', '11X,' 'RX', '11X,' 'RY', '11X,' 'RZ'/)
C
C ===========================================================
C C INCLUDE (PROCESS)
C SUBROUTINE OUT8(IOUT)
C IMPLICIT REAL*8 (A-H,O-Z)
C C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C C C DIRVEC

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
REAL*4 SHELLZ
COMMOS/INPUTS/NODES,HELEM,HEDF,HLINC,MAXI,IFLAG1,IFLAG2,IDIM,
1 MINDF,MNMDR,MFREE
COMMOS/INCR11/FRACT(10),HLINCI(10),LDOST,INCPTA
COMMOS/INPUTS/ISP(10000)
COMMOS/INPUTS/ISP(10000)
COMMOS/INPUTS/ISP(10000)
COMMOS/MAIN/RE(60000)
COMMOS/TRANS/DC(3,3)
COMMOS/SKTR2/SHELLZ(3,10000)
COMMOS/OUTPUT/IDEL(0000),IDUSD(10000)
DIMENSION DUMMY(6)

C IF (IDIM .EQ. 3) THEN
DO 110 K1 = 1, NODES
I = MINDF(K1)
ICODE = IAND(ISPB(K1),286)
ISP = IAND(ISPB(K1),4)
I = I + IDIM
IF (ICODE.GT.0 .AND. ISP .NE. 4) THEN
DC(1,3) = DBLE(SHELLZ(K1,1))
DC(2,3) = DBLE(SHELLZ(K1,2))
DC(3,3) = DBLE(SHELLZ(K1,3))
CALL DIRVEC (DC(1,1),DC(1,2),DC(1,3))
DO 100 K2 = 1, 3
CST = 0.
CST1 = 0.
IDIR = I + 1
CST = CST + RE(IDIR)*DC(K2,1)
IDIR = I + 2
CST = CST + RE(IDIR)*DC(K2,2)
IDIR = I + 3
CST = CST + RE(IDIR)*DC(K2,3)
DUMMY(K2) = CST
100 CONTINUE
C
DUMMY(K2) = DUMMY(K2)
IDIR = I + 1
RE(IDIR) = DUMMY(1)
IDIR = I + 2
RE(IDIR) = DUMMY(2)
IDIR = I + 3
RE(IDIR) = DUMMY(3)
ENDIF
110 CONTINUE
C
WRITE (IDOUT, 900)
DO 120 K1 = 1, NODES
IF (BTST(ISPB(K1),10)) THEN
ID = HEDP(K1)
IF(IDUSD(K1).EQ.1)WRITE(IDOUT,910)K1,(RE(ID+K2),K2=1,HEDF)
ENDIF
120 CONTINUE
C
RETURN
900 FORMAT(111,12X,'MODAL EQUILIBRIUM FORCES',
1 ' AND MOMENTS'/, 'NODE NO.',
2 'IIX','FX','IIX','FY','IIX','FZ','IIX','MX','IIX','MY','IIX','MZ'/)
910 FORMAT(111,12P,6E13.6)
END
SUBROUTINE OUTD(IGUT)

SUBROUTINE GETSTR ASSEMBLES THE GLOBAL STIFFNESS MATRIX AND/OR STORES THE NODE NUMBERS OF THE CORRECT ELEMENT AND THE POSITION OF THE ELEMENT MATRICES IN THE GLOBAL MATRICES.

II(J) = POSITION OF LOCAL STIFFNESS TERMS IN THE GLOBAL STIFFNESS MATRIX.

SKG(I) = GLOBAL STIFFNESS MATRIX IN THE CONDENSED FORM

SK(I,J) = ELEMENT STIFFNESS MATRIX

(SK IS COMPUTED BY SUBPROGRAM STIFEL)

IMPLICIT REAL*8 (A-E,O-Z)

SWITCHES: RESUM=100:10,FORMAT=900:10

SWITCHES:

C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

REWIR ELINFO ELISTM LYINFO ISH3DG ISH3ELE

SHERM DIRVEC ICGET JACB3D B3DLS GETSK

JACSEL BSEL BSHEL EQUILB

C ---------- CONDENSED DATA STORAGE IS ARRAY IFBODY.

INTEGER ELNUM

CHARACTER*47 CTEMP

REAL*4 SHELLZ,THICK

REAL*8 LTHICK,N,BXI,BXI,BXI

COMMON/LAYERB/LTHICK(9),ZS(3,3),ELAFFS,MATRL,LYRUM

COMMON/UTIL1/STRESS(9),STRAIN(9),STRELA(9),CTEMP

COMMON/ELST3/STAS(9)

COMMON/SHLDIR/VECT(3,3,9)

COMMON/INPUT/ISPE(100000)

COMMON/SKX12/SHELLZ(3,10000)

COMMON/ISHAP2/W(27)

COMMON/MAIB4/RE(60000)

COMMON/INPUT/THICK(9),IFLAG

COMMON/ISHAP1/H(20,27),XI(20,27),ETA(20,27),BXI(20,27),BI(20,27)

COMMON/INPUT/NODES,BELEM,BEPL,ELINC,MSK,IFLAG1,IFLAG2,IDIM,

1 NODES,NCOLOR,NFREE

COMMON/IFBC/DFACT(10),ELINC(10),LDCGET,INCPTR

COMMON/INPUT/MIPPI,MIPETA,MIPSI,MIP,MIPCD

COMMON/INPUT/BFP(20,6000)

COMMON/ASSEM2/ITI(120)

COMMON/FREEB/IFBCDY(10000)

COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDEEP,LDEV,LDEVS

COMMON/TRANS/DC(3,3)

C DIMENSION Dummy(6)

CALL Rewir
C ---- BIT STORAGE ORGANIZATION FOR IFBODY
C
C BIT RANGE ENTITY DESCRIPTION
C 1-15 ELEMENT 0; DOES NOT BELONG TO FREEBODY
C 16-30 NODES 0; DOES NOT BELONG TO FREEBODY
C FOR ELEMENTS FREEBODY NUMBER IS THE BIT NUMBER.
C FOR NODES FREEBODY NUMBER IS BIT NUMBER MINUS 15.
C MDOF IS THE MAXIMUM NUMBER OF DEGREES OF FREEDOM
C MDOF = NDF*NNODES
C
C ---- IFREE IS THE FREEBODY DIAGRAM NUMBER
C DO 220 IFREE = 1, NFREE
C
C ---- INITIALIZE THE EQUILIBRIUM LOAD VECTOR FOR EACH FREEBODY
C DO 100 K1 = 1, MDOF
R (K1) = 0.
100 CONTINUE
C
C MDOF = NUMBER OF COLUMNS IN THE B MATRIX
C NRB = NUMBER OF ROWS IN THE B MATRIX
C NNEL = NUMBER OF NODES IN THE ELEMENT
C DO 180 ELNUM = 1, HELEH
C CALL ELINFO (ELNUM, ITYPE, NNEL, IFLAG, ISTART, LINES)
CALL ELINTH (ELNUM, IDENT, INTCOD, NIPXI, NIPETA, NIPSI,
MATNUM, THICK)
C*VDIR: PREFER SCALAR
DO 110 K1 = 1, NNEL
I1 = NDF*(K1-1)
I2 = NDF*(NDF(K1,ELNUM)-1)
C*VDIR: PREFER SCALAR
DO 110 K2 = 1, NDF
K = I1 + K2
II(K) = I2 + K2
110 CONTINUE
120 CONTINUE
C DO 160 ELNUM = 1, NLAYRS
CALL LINFO (ELNUM, LTHICK, ZS, MATRL, DCPS, NEL,
ELNUM, NIPXI, NIPETA, NIPSI)
C
C NELEL = NUMBER OF ELEMENT NODEAL DEGREES OF FREEDOM
C NCB = NUMBER OF COLUMNS IN THE B MATRIX
C NRB = NUMBER OF ROWS IN THE B MATRIX
C ITYPE = ELEMENT TYPE
C IFLAG = ADDITIONAL IDENTIFIER FOR THE ELEMENT
C
IF (ITYPE .LE. 0) GO TO 170
IF (ITYPE.NE.0 .OR. IDENT.NE.0) THEN
IF (ITYPE .GT. 300) THEN
IF (IFLAG .EQ. 0) THEN

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\begin{verbatim}
HCB = 3*BBEL
SNB = 6
HENDF = 3
CALL ISBBDG (ITYPE, NSEL, IERROR)
ELSE IF (IFLAG .EQ. 4) THEN
  NCB = 6*BBEL
  SNB = 6
  HENDF = 6
  CALL ISBBDL (ITYPE, NSEL, IERROR)
ENDIF
ENDIF

C ITYPE1 = ITYPE
C IDENT1 = IDENT
C ---- FOR SHELLS EVALUATE THE LOCAL SHELL COORDINATES OF THE NODES
C
IF (BTEST(IFBODY(ELNUM),IFREE)) THEN
  IF (ITYPE.GT.500 .AND. IFLAG.EQ.4) THEN
    DO 130 K1 = 1, NSEL
    XP = ROP(K1,ELNUM)
    ICODE = IABD(ISPB(KP),4)
    C
    C ---- IF SHELL ROTATIONS ARE ASSEMBLED IN THE LOCAL SHELL COORDINATE
    C SYSTEM THEN RETRIEVE THE LOCAL Z-AXIS FROM STORAGE. ELSE
    C EVALUATE THE NORMAL TO THE SHELL MID PLANE BY A CALL TO THE
    C SHORM ROUTINE.
    C
    IF (ICODE .GT. 0) THEN
      CALL SHORM (ELNUM, K1, NSEL, ITYPE,
      1  VECT(1,1,K1),VECT(1,2,K1),VECT(1,3
      2 ,K1))
    ELSE
      VECT(1,3,K1) = DBLE(SHELLZ(1,KP))
      VECT(2,3,K1) = DBLE(SHELLZ(2,KP))
      VECT(3,3,K1) = DBLE(SHELLZ(3,KP))
      CALL DIRVEC (VECT(1,1,K1),VECT(1, 2,K1)
      1 ,VECT(1,3,K1))
    ENDIF
  130 CONTINUE
ENDIF

C C C ---- RETRIEVE STRESSES FROM STORAGE
C CALL IODET (LDEV1, 96, '(A96)', 5)
C C ---- CHECK TO SEE IF ELEMENT BELONGS TO FREEBODY DIAGRAM
C IF (BTEST(IFBODY(ELNUM),IFREE)) THEN
  DO 140 K1 = 1, 6
  STRS(K1) = STRESS(K1)
  CONTINUE
C C ---- GEOMETRICALLY LINEAR PROBLEMS
C IF (IFLAG1 .EQ. 0) THEN
  IF (ITYPE .GT. 500) THEN
  \end{verbatim}
IF (IFLAG .EQ. 0) THEN

C ------ EQUILIBRIUM VECTOR FOR 3D LINEAR SOLID ELEMENTS

CALL JAC3D (INTGDS, ELMU, NEEL, ITERMIN, DETJAC)
CALL B3DLS (NEEL)
CST = DETJAC*W(INTGDS)
ELSE IF (IFLAG .EQ. 4) THEN

C ------ EQUILIBRIUM VECTOR FOR 3D LINEAR SHELL ELEMENTS

C ------ VECTOR V13 RETURNED BY JACSEL IS NORMAL TO MID Surface OF THE
C SHELL AT INTEGRATION POINTS

CALL GETSHL (INTGDS, ELMU, NEEL, THICK, RAD)
ISET = 0
SIP = SI(INTGDS)
CALL JACSEL (INTGDS, ELMU, NEEL, THICK, DETJAC, V13, ISET, SIP)

C ------ THE COORDINATES VECTORS V11 AND V12 ARE EVALUATED BY DIRVEC
C WHICH IS PART OF THE ELEMENT LIBRARY MODULE

CALL DIRVEC (V11, V12, V13)
CALL BTSHEL (ELMU, NEEL, 6, NEEL, 3, 2)
ENDIF
ENDIF

150 CONTINUE
160 CONTINUE
170 CONTINUE
180 CONTINUE

IF (IDIM .EQ. 3) THEN
DO 200 K1 = 1, NODES
IF (BTEST(IFBODY(K1),IFREE+16)) THEN
I = NEEL*(K1-1)
ICODE = IAND(ISPB(K1),256)
ISPB = IAND(ISPB(K1),4)
I = I + IDIM
IF (ICODE .GT. 0 .AND. ISPB .NE. 4) THEN
DC(1,3) = DBLE(SHELLZ(1,K1))
DC(2,3) = DBLE(SHELLZ(2,K1))
DC(3,3) = DBLE(SHELLZ(3,K1))
CALL DIRVEC (DC(1,1),DC(1,2),DC(1,3))
DO 190 K2 = 1, 3
CST = 0.
CSTI = 0.
IDIR = I + 1
CST = CST + RE(IDIR)*DC(K2,1)
IDIR = I + 2
CST = CST + RE(IDIR)*DC(K2,2)
IDIR = I + 3
CST = CST + RE(IDIR)*DC(K2,3)
DUMMY(K2) = CST
190 CONTINUE
ENDIF
ENDIF

C

IDIR = I + 1
\begin{verbatim}
RE(IDIR) = DUMMY(1)
IDIR = I + 2
RE(IDIR) = DUMMY(2)
IDIR = I + 3
RE(IDIR) = DUMMY(3)
ENDIF
ENDIF
CONTINUE
ENDIF
C
WRITE (IDUT, 900) IFREE
IF16 = IFREE + 16
DO 210 K1 = 1, NHODES
IF (BTSTE(IYPBDY(K1), IF16)) THEN
    IF (BTSTE(ISPB(K1), 10)) THEN
        ID = BWDF*(K1-1)
        WRITE (IDUT, 910) K1, (RE(ID+K2), K2 = 1, BWDF)
    ENDIF
ENDIF
210 CONTINUE
C
CALL REWIN
220 CONTINUE
C
RETURN
900 FORMAT(/I1H1,20X,'M D D A L E Q U I L I B R I U M F O R C E S ',
   1 ' A N D M O M E N T S '/20X,'F O R F R E E B O D Y ',
   2 ' D I A G R A M , '/18/'H O D A L E Q U I L I B R I U M F O R C E S ' ,
   3 ' A N D M O M E N T S '/20X,'F O R F R E E B O D Y ',
   4 ' D I A G R A M '/18)
910 FORMAT(I11,1P,6E13.6)
END
C
C***************************************************************************
C***************************************************************************
C INCLUDE (PROCESS)
C SUBROUTINE OUTIO(IOUT)
C IMPLICIT REAL*8 (A-H,O-Z)
C . . .SWITCHES: REWIN=100:10,FORMAT=900:10
C . . .SWITCHES:
C***************************************************************************
C***************************************************************************
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C C N I V E C
C
C***************************************************************************
C***************************************************************************
C REAL=4 SHELLZ
INTEGER ELBN
COMMON/INPUTS/NODES,SELEN,REDF,BLINC,MBIT,IFLAG1,IFLAG2,IDIM,1
NNODE,NCOLOR,IFREE
COMMON/TCFIC1/FRATC(10),BLINC1(10),LDCONT,INCPTO
COMMON/INS/ISP(10000)
COMMON/DEVICE/LDEV,LDEV2,LDEV4,LDEP,LDEP2,LDEP4,LDEP6,LDEV5,LDEVST
COMMON/MAB4/RE(60000)
COMMON/OUPT2/IOCL(8000),IOMBD(10000)
COMMON/TRANS/DCC(3,3)
COMMON/STRKZ/SHELLZ(3,10000)
DIMENSION DUMMY(6)
C
C
IF (IDIM .EQ. 3) THEN
   DO 110 K1 = 1, NHODES
   
\end{verbatim}
C ---- TEST BIT 18 OF ISPB TO SEE IF THE NODE IS A SUPPORT
C
IF (BTEST(ISPB(K1),18)) THEN
  I = NBF+K1-1
  ICODE = IAND(ISPB(K1),256)
  ISPS = IAND(ISPB(K1),4)
  I = I + IDIM
  IF (ICODE.GT.0 .AND. ISPS.NE.4) THEN
    DC(1,3) = DBLE(SHELLZ(1,K1))
    DC(2,3) = DBLE(SHELLZ(2,K1))
    DC(3,3) = DBLE(SHELLZ(3,K1))
    CALL DIRVEC (DC(1,1),DC(1,2),DC(1,3))
    DO 100 K2 = 1, 3
      CST = 0.
      CST1 = 0.
      IDIR = I + 1
      CST = CST + RE(IDIR)*DC(K2,1)
      IDIR = I + 2
      CST = CST + RE(IDIR)*DC(K2,2)
      IDIR = I + 3
      CST = CST + RE(IDIR)*DC(K2,3)
    END DO
    DUMMY(K2) = CST
  CONTINUE
  I = I + 1
  RE(IDCIR) = DUMMY(1)
  IDIR = I + 2
  RE(IDCIR) = DUMMY(2)
  IDIR = I + 3
  RE(IDCIR) = DUMMY(3)
ENDIF
100 CONTINUE
ENDIF
C
END
C
WRITE (IDUT, 900)
DO 120 K1 = 1, NBODES
  IF (BTEST(ISPB(K1),18)) THEN
    ID = NBF+K1-1
  ELSE IF (IDMOD(K1).EQ.1) WRITE(IDUT,910)K1,(RE(ID+K2),K2=1,NBDF)
  END IF
120 CONTINUE
C
RETURN
900 FORMAT(/1H1,10X,'SUPPORT REACTION FORCES',/1H1,10X,'M M MENTS',/1H1,10X,'NODE NO.',/9X,'F1',/9X,'F2',/9X,'F3',/9X,'M1',/9X,'M2'/)
910 FORMAT(8I1,1P,6E13.5)
END
C
C ****************************************************** D U T 11 ******************************************************
C
C INCLUDE (PROCESS)
C Subroutine OUT11(IOUT)
C implicit real*8(A-H,O-Z)
C...SWITCHES: RESUMB=100:10,FORMAT=900:10
C...SWITCHES:
C***********************************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
REAL*4 YZ
COMMON/INPUT3/XYZ(3,10000)
COMMON/INPUT6/NUMDES,ELEM,BNDF,SLISC,MSIT,IFLAG1,IFLAG2,IDIM,
1 NODE,BCOLOR,NFREE
COMMON/EXTRP/FRACT(10),ELISCI(10),LDCONT,IECPTR
COMMON/INPUT8/ISP8(10000)

ID = 0.

WRITE (IOUT, 900)
DO 100 K1 = 1, NUMDES
  IF (TEST(ISPB(K1),10)) WRITE (IOUT, 910) K1, ID, XYZ(K1),
1 1 XYZ(2,K1), XYZ(3,K1)
100 CONTINUE
RETURN
900 FORMAT(/10H20X,'H O D A L  C O O R D I N A T E S'/
1 2X,'N O D E  I D . ',5X,'C O O R D  N O . ',12X,'X',12X,'Y',12X,'Z'/)
910 FORMAT(10,I4,1P,6E13.5)
END

C=====================================================================
CINCLUDE (PROCESS)

SUBROUTINE EXTRP(VALUE)

IMPLICIT REAL*8 (A-E, O-Z)

C...SWITCHES: REUN/NUM=100:10,FORMAT=900:10
C...SWITCHES:
C=====================================================================
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELISFI  ELIMTH  LAGR61 JGET  RENB
C
C=====================================================================

REAL*4 THICK
REAL*4 YIELD
INTEGER ELNUM
COMMON/UTIL1/STRESS(6),STRAIN(6),STRELA(6),CENTER(6),WORK,YIELD
COMMON/INPUT1/EIPX1,EIPS1,EIPS2,EITCD
COMMON/INPUT2/BOP(20,6000)
COMMON/INPUT6/ISP8(10000)
COMMON/INPUT7/UNITPE(10)
COMMON/DEVICE/LSBV1,LDEN,LDEN3,LDE4,LDEN6,LDEN7,LDEN8,LDEN9,LDEN10,
COMMON/GRAPE/SYS(12000),IYV(12000),ILS(12000),IHS(12000)
COMMON/ITRP1/ITREP(12000),ILREP(12000)
COMMON/EXTRP/IET38(9),IET22(4)
COMMON/INPUT8/NUMDES,ELEM,BNDF,SLISC,MSIT,IFLAG1,IFLAG2,IDIM,
1 NODE,BCOLOR,NFREE
COMMON/EXTRP1/FRACT(10),ELISCI(10),LDCONT,IECPTR
COMMON/INPUT9/THICK(K1),IFLAG
COMMON/ISP9/THICK(K1),IFLAG
DIMENSION VALUE( * ),STRB(6,64),STRS(6,64),CAUC(6,64),
1 "CAUCH(6),AWORK(64)
C
C
DO 100 K1 = 1, 14*NUMDES
  VALUE(K1) = 0.
100 CONTINUE
C
C
C
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
ITYPE1 = 0
IDENT1 = 0
IEND = 0
DO 250 ELNUM = 1, NELEM
   CALL ELISFC (ELNUM, ITYPE, BBEL, IFLAG, ISTART, LINES)
   CALL ELISTM (ELNUM, IDENT, INTCOD, HIPXI, HIPETA, HIPSI, MATNUM, THICK)
C
   IF (ITYPE .LE. 0) GO TO 240
   IF (ITYPE .NE. ITYPE1 .OR. IDENT .NE. IDENT1) THEN
      CALL LAGRGl (ITYPE, IFLAG, BBEL, IERROR)
      IF (ITYPE .GT. 300) THEN
         IEND = 6
      ELSE
         IEND = 4
      ENDIF
   ENDFN
   ITYPE1 = ITYPE
   IDENT1 = IDENT
C
   DO 120 INTGPH = 1, HIP
      IF (MATYPE(MATNUM) .EQ. 1) THEN
         CALL IDIGET (LDEV1, 96, '((A96)', 5)
      ELSE
         CALL IDIGET (LDEV1, 201, '((A201)', 6)
         AWORK(INTGPH) = WORK
      ENDIF
   C
   DO 110 K1 = 1, IEND
      STRS(K1,INTGPH) = STRESS(K1)
      STRA(K1,INTGPH) = STRAIN(K1)
110  CONTINUE
120  CONTINUE
C
   DO 150 K1 = 1, BBEL
      NODE = RIDP(K1, ELNUM)
      DO 140 K2 = 1, IEND
         ID = (K2-1)*NBODES + NODE
         DO 130 K3 = 1, HIP
            VALUE(ID) = VALUE(ID) + STRS(K2,K3)*P(K3,K1)
130  CONTINUE
140  CONTINUE
150  CONTINUE
C
   DO 180 K1 = 1, BBEL
      NODE = RIDP(K1, ELNUM)
      DO 170 K2 = 1, IEND
         ID = (IEND+K2-1)*NBODES + NODE
         DO 160 K3 = 1, HIP
            VALUE(ID) = VALUE(ID) + STRA(K2,K3)*P(K3,K1)
160  CONTINUE
170  CONTINUE
180  CONTINUE
C
   DO 210 K1 = 1, BBEL
      NODE = RIDP(K1, ELNUM)
      DO 200 K2 = 1, IEND
         ID = (2*IEND+K2-1)*NBODES + NODE
         DO 190 K3 = 1, HIP
            VALUE(ID) = VALUE(ID) + CAU(C(K2,K3)*P(K3,K1)
190  CONTINUE
200  CONTINUE
210  CONTINUE
  C
  C  ID1 = 3*IEND+NBODES
  C  DO 360  K1 = 1 , NBEL
  C  ID = ID1 + N0P(K1 , ELNUM)
  C  DO 360  K3 = 1 , NIP
  C  VALUE(ID) = VALUE(ID) + VOLUMS(K3 )*P(K3, K1)
  C360  CONTINUE
  C
  C  ID1 = (3*IEND+1)*NBODES
  DO 230  K1 = 1 , NBEL
    ID = ID1 + N0P(K1 , ELNUM)
    DO 220  K3 = 1 , NIP
      VALUE(ID) = VALUE(ID) + AWORK(K3)*P(K3 , K1)
  220  CONTINUE
  230  CONTINUE
  240  CONTINUE
  250  CONTINUE
  C
  DO 270  K2 = 1 . 14
    ID1 = (K2-1)*NBODES
    DO 260  NODE = 1 . HNODE
      ITEST = IAND(ISPB(NODE) , 1024)
      IF ( ITEST .GT. 0 ) THEN
        IRNODE = IREP(NODE) / 32
        ID = ID1 + IRNODE
        VALUE(ID) = VALUE(ID)/DFLOAT(IRNODE)
      ENDIF
  260  CONTINUE
  270  CONTINUE
  C
  CALL REW1
  280  CONTINUE
  RETURN
  END

C
C ***************************************************************
C
C INCLUDE (PROCESS)
C
SUBROUTINE LA GRG I (ITYPE , IFLAG ,NBEL , ERROR)
C
C ***************************************************************
C
C PRO G R A M :
C
C LAGRG I EVALUATES THE LAGRANGE POLYNOMIAL COEFFICIENTS
C WHICH ARE USED FOR INTERPOLATING VALUES FROM GAUSSIAN
C INTEGRATION POINTS TO THE NODES. THIS SUBROUTINE MAY BE
C USED FOR QUADRILATERAL AND HEXAHEDRAL ISOPARAMETRIC ELEMENTS
C WHICH USE ANY ORDER OF GAUSS INTEGRATION.
C
C A R G U M E N T L I S T:
C
C ITYPE = INTERNAL ELEMENT TYPE NUMBER.
C IFLAG = ANALYSIS TYPE FLAG.
C NBEL = NUMBER OF NODES IN THE ELEMENT.
C
C D E F A U LT R E T U R N:
C
C ARRAY P IS COMMON ISPCD1 CONTAINS THE LAGRANGE INTERPOLATION
C POLYNOMIALS.
C
Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE

GAUSS

REAL*4 XI, ETA, XI, SI, ETA
REAL*8 P, XI, ETA, SI, PSI, W, XI, W, ETA, WSI
COMMON/IP/IPIX, IPETA, HIPSI, HIP, HIPCOD
COMMON/ISP/PI/7, 7, 7)

--- ELLIB1 CONTAINS THE COORD. OF THE NODES FOR HEXAHERAL
ELEMENTS IN THE NATURAL (ISOPARAMETRIC) COORDINATE SYSTEM.
These VALUES ARE SINGLE PRECISION REAL NUMBERS
COMMON/ELLIB1/XI20), ETAI (20)
DIMENSION XI(4), ETAI(4), SI(4), PX1(4), PETA(4), PSI(4)

--- ELLIB2 CONTAINS THE COORD. OF THE NODES FOR QUADRILATERAL
ELEMENTS IN THE NATURAL (ISOPARAMETRIC) COORDINATE SYSTEM.
These VALUES ARE SINGLE PRECISION REAL NUMBERS
COMMON/ELLIB2/XI9), AETA(9)
DIMENSION XI(4), ETA(4), SI(4), PX1(4), PETA(4), PSI(4)

--- NEXT IF BLOCK EVALUATES THE LAGRANGE POLYNOMIALS FOR THE
HEXAHEDRAL OR SHELL QUADRILATERAL ELEMENTS.

IF (ITYPE .GT. 300) THEN
CALL GAUSS (HIP XI, WX, XI)
CALL GAUSS (HIP ETA, WETA, ETA)
CALL GAUSS (HIP SI, WSI, SI)
NIP = NTPXI*HIP ETA#NIPSI
I = NIP XI*NIP ETA

--- EVALUATES THE LAGRANGE POLYNOMIALS FOR HEXAHEDRAL ELEMENTS.

IF (IFLAG .EQ. 0) THEN
DO 190 NODE = 1, HEL
XIN = XIICNODE)
ETAN = ETAI(HODE)
SIN = SI(HODE)

DO 110 ISI = 1, NIPSI
PSI(ISI) = 1.0
DO 100 Kl = 1, HIPSI
IF (Kl .NE. ISI) PSI(ISI) = PSI(ISI)*(SIN-SI(KI))/(SI(ISI)-SI(KI))
100 CONTINUE
110 CONTINUE

DO 130 IETA = 1, HIPETA
PET(A(IETA)) = 1.0
DO 120 K1 = 1, HIPETA
IF (K1 .GE. IETA) PET(A(IETA)) = PET(A(IETA))**(1
```fortran
C Evaluate the polynomials for shell elements. Loop 110 assigns a SI value to the natural coordinates of the nodes. SI equals to -1., 0., and 1. correspond to the bottom surface, mid-surface, and top surface of the shell elements.
C
ELSE IF (IFLAG .EQ. 4) THEN
C
SIX = -2.DO
DO 300 ISURF = 1, 3
SIN = SIN + 1.0DO
DO 290 NODE = 1, BKEL
XIN = AXI(NODE)
ETAN = ETA(NODE)
DO 210 ISI = 1, HIPSI
PSI(ISI) = 1.0
C WRITE(*,*)'ISI',ISI,'SI',SI(ISI)
DO 200 K1 = 1, HIPSI
IF (K1 .NE. ISI) PSI(ISI) = PSI(ISI)*(SIN-SI(K1))/(SI(ISI)-SI(K1))
200 CONTINUE
210 CONTINUE
C
DO 230 IETA = 1, HIPETA
PETA(IETA) = 1.0
DO 220 K1 = 1, HIPETA
IF (K1 .NE. IETA) PETA(IETA) = PETA(IETA)*((ETAN-ETA(K1))/(ETA(IETA)-ETA(K1)))
220 CONTINUE
230 CONTINUE
C
DO 250 IXI = 1, HIPXI
PXi(IXI) = 1.0
DO 240 K1 = 1, HIPXI
IF (K1 .NE. IXI) PXi(IXI) = PXi(IXI)*(XIN-XI(K1))/(XI(IXI)-XI(K1))
240 CONTINUE
250 CONTINUE
C
ETAN-ETA(K1))/(ETA(IETA)-ETA(K1))
```

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
DO 270 IETA = 1, NIPETA
   DO 260 III = 1, HIPXI
      INTGPB = (ISI-1)*I + (IETA-1)*NIPXI+IXI
      P(IBTPN,HODE) = PSIC(ISI)*PETA(IETA)*<
      PXKIII)
   260 CONTINUE
  270 CONTINUE
  280 CONTINUE
  290 CONTINUE
  300 CONTINUE
ENDIF
ENDIF
RETURN
END
C
C ========================= REVIEW =========================
C INCLUDE (PROCESS)
SUBROUTINE REVIEW(IDOF)
C
C PROGRAM:
C I)
C I REVIEW CHECKS THE VALIDITY OF NODE AND ELEMENT DEFINITIONS. IT
C I ALSO GENERATES AUTO
C I RESTRAINT INFORMATION FOR THE DRILLING DEGREES OF FREEDOM OF
C I SHELL ELEMENTS. THE ROTATIONAL DEGREES OF FREEDOM OF THE NODES
C I WHICH ARE SHARED BY SHELL ELEMENTS WITH THEIR MID-SURFACE
C I NORMALS PARALLEL TO EACH OTHER ARE ASSEMBLED IN THE LOCAL
C I COORDINATE SYSTEM OF THE NODE. THE
C I Z-PRIME AXIS OF THE LOCAL COORDINATES IS NORMAL TO THE SURFACE
C I OF THE SHELL. SINCE THERE IS NO STIFFNESS CONTRIBUTION TO THE
C I ROTATIONAL DEGREE OF FREEDOM ABOUT THE Z-PRIME, THIS DEGREE OF
C I FREEDOM IS EXPLICITLY ELIMINATED BY SETTING THE APPROPRIATE
C I TERM IN THE IDOF ARRAY EQUAL TO 2. FOR NODES THAT ARE
C I RESTRAINED BY THE USER THE ROTATIONAL DEGREES OF FREEDOM IS
C I ASSEMBLED IN THE GLOBAL COORDINATE SYSTEM. IN THIS SITUATION
C I WHEN THE USER DEFINED RESTRAINED ROTATIONS HAVE NO COMPONENTS
C I IN THE DIRECTION OF THE DRILLING DEGREE OF FREEDOM THE DRILLING
C I DEGREES OF FREEDOM WILL BE RESTRAINED UNLESS THE NODE IS SHARED
C I BY AT LEAST TWO OUT OF PLANE SHELL ELEMENTS. ALL NODES THAT ARE
C I SHARED BY NON-TANGENT SHELL ELEMENTS WILL HAVE THEIR ROTATIONAL
C I DEGREES OF FREEDOM ASSEMBLED IN THE GLOBAL REFERENCE FRAME.
C I FOR THESE NODES SOME STIFFNESS EXISTS FOR ALL ROTATIONAL
C I DEGREES OF FREEDOM, HENCE AUTO RESTRAINT IS NOT INVOKED.
C I II)
C I IT DEFINES THE NODES THAT ARE RELATED TO FREEBODY DIAGRAMS.
C I ON ENTRY:
C I IDOF = IS THE ARRAY CONTAINING THE NODAL RESTRAINT INFORMATION
C I SPECIFIED BY THE USER.
C I 0; FREE NODE
C I 1; RESTRAINED WITH ZERO DISPLACEMENT OR ROTATION
C I -1; RESTRAINED WITH NONZERO DISPLACEMENT OR ROTATION
C I ON RETURN:
C I IDOF = IS THE ARRAY CONTAINING THE MODAL RESTRAINT INFORMATION
C I SPECIFIED BY THE USER AND BY THE SHELL AUTO RESTRAINS.
C I 0; FREE NODE
C I 1; RESTRAINED WITH ZERO DISPLACEMENT OR ROTATION
C I -1; RESTRAINED WITH NONZERO DISPLACEMENT OR ROTATION
C I 2; RESTRAINED WITH ZERO ROTATION (AUTO RESTRAIN)
C I
C I******************************************************************************
C I
C IMPLICIT REAL*8 (A-H,O-Z)
C . . .SWITCHES: REHUHB=100:10,FORMAT=900:10
C . . .SWITCHES:
C******************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C ELINFO ELIHTM ERRORS SHROM DIRCOS DOTPRC
C
C******************************************************************************
C
INTEGER ELNUN
CHARACTER*6 COMM
CHARACTER*7 CELEM,CHODE
CHARACTER*80 BUFF,BUFFER
REAL*4 XYZ,XAXIS,YAXIS,THICK
COMMON/INPUT2/BDP(20,6000)
COMMON/INPUT3/XYZ(3,10000)
COMMON/INPUT6/BNODES,HELEM,REDF,MLINC,MBIT,IFLAG1,IFLAG2,IDIIM,
1 NNODE,ECOLOR,RFREE
COMMON/INCR11/FRAC(10),MLINC(10),LDCOFT,INCPTR
COMMON/INPUT4/XAXIS(4,10000),YAXIS(4,10000)
COMMON/EXTR2/SHLLE(3,10000)
COMMON/TRANS/DC(3, 3)
COMMON/INPUTE/ISP8(10000)
COMMON/COMP2/COMM,BUFFER,BUFF
COMMON/FREE/IFBODY(10000)
COMMON/INPUT9/THICK(8),IFLAG
DATA CELEM,CHODE/"ELEMENT","NODE"/
C ------ TSPB(K) STORES A SERIES OF CRITICAL INFORMATION ABOUT THE
C NODES AND ELEMENTS. THE BIT STORAGE FOR EACH ELEMENT OF THIS
C ARRAY IS AS FOLLOWS.
C
C BIT 0 LOCAL COORDINATE SYSTEM FOR TRANSLATIONS
C BIT 1 LOCAL COORDINATE SYSTEM FOR ROTATIONS IS DEFINED
C BIT 2 0; SHELL ROTATIONS IN LOCAL SHELL COORDINATE SYSTEM
C 1; SHELL ROTATIONS ARE IN GLOBAL COORDINATE SYSTEM
C BIT 3 SIGN BIT FOR THE THIRD DIRECTION COSINE OF THE
C LOCAL TRANSLATION X-AXIS (0=+,1=-).
C BIT 4 SIGN BIT FOR THE THIRD DIRECTION COSINE OF THE
C LOCAL TRANSLATION Y-AXIS (0=+,1=-).
C BIT 5 SIGN BIT FOR THE THIRD DIRECTION COSINE OF THE
C LOCAL TRANSLATION Z-AXIS (0=+,1=-).
C BIT 6 SIGN BIT FOR THE THIRD DIRECTION COSINE OF THE
C LOCAL ROTATION Y-AXIS (0=+,1=-).
C BIT 7 SIGN BIT FOR THE THIRD DIRECTION COSINE OF THE
C LOCAL SHELL ROTATION Z-AXIS (0=+,1=-).
C BIT 8 0; NODE IS NOT SHARED WITH ANY SHELL ELEMENTS
C 1; NODE IS SHARED WITH AT LEAST ONE SHELL ELEMENT
C BIT 9 0; NODE IS NOT SHARED WITH ANY ELEMENTS
C 1; NODE IS SHARED WITH AT LEAST ONE ELEMENT

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
BIT 10 0; POSITION OF THE NODE HAS NOT BEEN DEFINED
1; POSITION OF THE NODE HAS BEEN DEFINED

BITS 1,0; CARTESIAN LOCAL TRANSLATION COORD. SYSTEM
11, 12 0,1; CYLINDRICAL LOCAL TRANSLATION COORD. SYSTEM
1,1; SPHERICAL LOCAL TRANSLATION COORDINATE SYSTEM.

BITS 1,0; CARTESIAN LOCAL ROTATION COORD. SYSTEM
13, 14 0,1; CYLINDRICAL LOCAL ROTATION COORD. SYSTEM
1,1; SPHERICAL LOCAL ROTATION COORDINATE SYSTEM.

BITS 1,0; CARTESIAN DEFINITION COORD. SYSTEM
15, 16 0,1; CYLINDRICAL DEFINITION COORD. SYSTEM
1,1; SPHERICAL DEFINITION COORDINATE SYSTEM.

BIT 17 0; NODE IS NOT AN INTERFACE NODE.
1; NODE IS AN INTERFACE NODE.

BIT 18 0; NODE IS NOT A SUPPORT
1; NODE IS A SUPPORT

BIT 20 0; ELEMENT CONNECTIVITY HAS NOT BEEN DEFINED
1; ELEMENT CONNECTIVITY HAS BEEN DEFINED

DIMENSION IDOF(*),V1(3),V2(3),V3(3)

THE DEFAULT TOLERANCE ANGLE IS 3 DEGREES

TOLCOS IS COSINE OF THE TOLERANCE ANGLE
ANGLE IS THE TOLERANCE ANGLE IN RADIANS

ANGLE = 3.0*0.017453292
TOLCOS = DCOS(ANGLE)
DO 170 ELHUH = 1, HELEM
CALL ELIFUG (ELUM, ITYPE, ISEL, IFLAG, ISTART, LINES)
CALL ELIFMT(ELUM,IDENT,LIFCOD,HIPXI,HIPETA,HIPSI,MAT,THICK)
IF (ITYPE .LE. 160) GO TO 160

C --- TEST BIT 20 OF ISPB TO SEE IF THE ELEMENT CONNECTIVITY
C HAS BEEN DEFINED.
C
ITEST = IAND(ISPB(ELEM),1046576)
IF (ITEST .EQ. 0) THEN
  WRITE (BUFFER, *) 'CELEM, ELHUM'
  CALL ERRORS (14, 0, 'REVIEW')
ENDIF
C
IF (ITYPE .GT. 300) THEN
  IF (IFLAG .EQ. 0) THEN
    DO 100 K1 = 1, NSEL
      NODE = BOP(K1,ELUM)
    END
    ELSE IF (IFLAG .EQ. 4) THEN
      DO 140 K1 = 1, NSEL
    ELSE IF (IFLAG .EQ. 0) THEN
      DO 140 K1 = 1, NSEL
    END
ENDIF
C
C --- TEST BIT 10 OF ISPB TO SEE IF THE NODE USED TO DEFINE
C CONNECTIVITY HAS BEEN DEFINED.
C
ITEST = IAND(ISPB(NODE),1024)
IF (ITEST .EQ. 0) THEN
  WRITE (BUFFER, *) 'CELEM, NODE, CELEM, ELEM'
  CALL ERRORS (11, 0, 'REVIEW')
ENDIF
C
ISPB(NODE) = IBSET(ISPB(NODE),9)
100 CONTINUE
ELSE IF (IFLAG .EQ. 4) THEN
  DO 140 K1 = 1, NSEL
ENDIF
C
C --- TEST BIT 10 OF ISPB TO SEE IF THE NODE USED TO DEFINE

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CONNECTIVITY HAS BEEN DEFINED.

RODE = M0P(X1, ELUM)
ITEST = IMBDS(IPSPB(RODE), 1024)
IF (ITEST .EQ. 0) THEN
  WRITE (BUFFER, *) RODE, NODE, CELEM, ELUM
  CALL ERRORS (11, 0, 'REVIEW')
ENDIF

CALL SHERM (ELUM, X1, SHEL, ITYPE, V1, V2, V3)

----- SET BIT 8 OF THE ISPB ARRAY TO 1 (NODE IS SHARED BY A SHELL)
ISPB(RODE) = IBSET(ISPB(RODE), 8)
ISPB(HODE) = IBSET(ISPB(HODE), 9)

ID = BNDF*(RODE-1)

ISPS = IAHD(ISPB(HODE), 4)
ICODE = IAHD(ISPB(HODE), 2)

THE FOLLOWING IF BLOCK IS USED TO DETERMINE IF ANY ROTATIONAL
RESTRAINTS ARE ASSOCIATED WITH THE NODE. IF RESTRAINTS ARE
PRESENT, THE SHELL ROTATIONS WILL HAVE TO BE ASSEMBLED IN THE
GLOBAL COORDINATE SYSTEM. IT SHOULD THEN BE CHECKED TO SEE IF
THE DRILLING DEGREE OF FREEDOM OF THE SHELL IS RESTRAINED
PROPERLY. THE ABOVE PROCESS IS NOT REQUIRED IF IT HAS PREVIOUSLY
BEEN DETERMINED THAT THE SHELL ROTATIONS SHOULD BE ASSEMBLED
IN THE GLOBAL REFERENCE FRAME.

IF (ISPS .EQ. 0) THEN
  DO 110 K2 = 4, BNDF
    ID1 = ID + K2
    IF (IDOF(ID1).EQ.1 .OR. IDOF(ID1).EQ. (-1) ) ISPS = 4
  110 CONTINUE
ENDIF

IF (ISPS .EQ. 4) THEN
  I = 0
  DO 120 K2 = 4, BNDF
    I = I + 1
    ID1 = ID + K2
    IF (IDOF(ID1) .EQ. 0) THEN
      IF (ICODE .EQ. 0) THEN
        IF (DABS(V3(D)) .GT. TOLCOS) IDOF(ID1) = 2
        ELSE
          CALL DIRCOS (ICODE, IDIH, HODE)
          DOT = DOTPRV(V3, DC(1, I), 3)
          IF (DABS(DOT) .GT. TOLCOS) IDOF(ID1) = 2
        ENDIF
      ENDIF
    ELSE
      CALL DOPRO(V3, DC(1, I), 3, DDOT)
      IF (DABS(DDOT) .GT. TOLCOS) IDOF(ID1) = 2
    ENDIF
  120 CONTINUE
ENDIF

IF MORE THAN ONE SHELL SHARE THE NODE, THEN THE SUM OF THE
NORMAL VECTORS IS ACCUMULATED IN SHELLZ ARRAY. THE DOT PRODUCT
OF THE CURRENT SHELL NORMAL V3 AND THE APPROPRIATE SHELLZ
COMPONENTS WILL DETERMINE IF THE CURRENT SHELL IS COPLAR.
(TANGENT) TO THE PREVIOUSLY PROCESSED SHELLS. IF SHELLS ARE NOT
COPLAR THEN SUFFICIENT ROTATIONAL STIFFNESS EXISTS IN ALL
DIRECTIONS. HENCE, ALL THE IDOF COMPONENTS RESTRAINED BY THIS
MODULE (IDOF(8)=2) SHOULD BE SET EQUAL TO ZERO.

C
C
C

COSXZP = DBLE(SHELLZ(1, NODE))
COSYZP = DBLE(SHELLZ(2, NODE))
COSZZP = DBLE(SHELLZ(3, NODE))

C

DOT = V3(1)**COSXZP - COSYZP*V3(2) + V3(3)**COSZZP
CHORM = DSQRT(COSXZP**2+COSYZP**2+COSZZP**2)
IF (CHORM .NE. 0.) THEN
BCOS = DOT/CHORM
ELSE
BCOS = 1.
ENDIF

C

IF (ISPS .NE. 4) THEN
IF (DABS(BCOS) .GE. TOLCOS) THEN
IF (IDOF(ID+6) .EQ. 0) IDOF(ID+6) = 2
ELSE
ISPS = 4
IF (IDOF(ID+6) .EQ. 2) IDOF(ID+6) = 0
ENDIF
ELSE
IF (DABS(BCOS) .LT. TOLCOS) THEN
DO 130 K2 = 4, 6
IDI = ID + K2
IF (IDOF(IDI) .EQ. 2) IDOF(IDI) = 0
130 CONTINUE
ENDIF
ENDIF
C

IF (ISPS .GT. 0) THEN
ITEMP = IBSET(ISPB(NODE),2)
ISPB(NODE) = ITEMP
ENDIF

C

SHELLZ(1, NODE) = SHELLZ(1, NODE) + V3(1)
SHELLZ(2, NODE) = SHELLZ(2, NODE) + V3(2)
SHELLZ(3, NODE) = SHELLZ(3, NODE) + V3(3)

C... NORMALIZE THE VECTOR SHELLZ.
C

SNORM = SQRT(SHELLZ(1, NODE)**2+SHELLZ(2, NODE)**2+SHELLZ(3, NODE)**2)
SHELLZ(1, NODE) = SHELLZ(1, NODE)/SNORM
SHELLZ(2, NODE) = SHELLZ(2, NODE)/SNORM
SHELLZ(3, NODE) = SHELLZ(3, NODE)/SNORM

140 CONTINUE
ENDIF
ELSE IF (ITYPE .GT. 200) THEN
DO 150 K1 = 1, SHEL
NODE = S0P(K1, ELNUM)
C
C ---- TEST BIT 10 OF ISPB TO SEE IF THE NODE USED TO DEFINE
C CONNECTIVITY HAS BEEN DEFINED.
C

ITEST = IAND(ISPB(NODE),1024)
IF (ITEST .EQ. 0) THEN
WRITE (BUFFER, *) CNODE, NCDE, CELEM, ELNUM
CALL ERRORS (11, 0, 'REVIEW')
ENDIF
C
ISPB(NODE) = IBSET(ISPB(NODE),9)
160 CONTINUE
ENDIF
160 CONTINUE
170 CONTINUE
C ---- IF A NODE IS NOT SHARED BY ANY SHELL ELEMENTS THEN RESTRAIN
C THE ROTATIONAL DEGREES OF FREEDOM.
C
DO 190 NODE = 1, NODES
IDEBT1 = IAND(ISPB(NODE),612)
IDEBT2 = IAND(ISPB(NODE),256)
190 CONTINUE
C ---- IF A NODE IS NOT SHARED BY ANY ELEMENT THEN RESTRAIN ALL DOF
C
IF (IDEBT1 .EQ. 0) THEN
  ID = NNDF*(NODE-1)
  DO 180 K1 = 1, NNDF
    IDOF(ID+K1) = 1
  180 CONTINUE
C ---- IF A NODE IS NOT SHARED BY ANY SHELL ELEMENTS THEN RESTRAIN
C THE ROTATIONAL DEGREES OF FREEDOM.
C
ELSE IF (NNDF.EQ.0 .AND. IDEBT2.EQ.0) THEN
  ID = NNDF*(NODE-1)
  IDOF(ID+3) = 1
  IDOF(ID+6) = 1
  CALL UNITVS (SHELLZD, BODE) , 3)
ENDIF
C ---- NORMALIZE THE SHELLZ VECTOR.
C
190 CONTINUE
C* DO 70 K1 = 1 , NODES
C* ID = NNDF*(K1-1)
C* C
C ---- READ AND GENERATE FREEBODY INFORMATION FOR NODES
C ---- CONDENSED DATA STORAGE IN ARRAY IFBODY.
C ---- BIT STORAGE ORGANIZATION FOR IFBODY
C
BIT RANGE ENTITY DESCRIPTION
C
1-16 ELEMENT 0; DOES NOT BELONG TO FREEBODY
C 1; BELONGS TO FREEBODY
C 16-30 NODES 0; DOES NOT BELONG TO FREEBODY
C 1; BELONGS TO FREEBODY
C
FOR ELEMENTS FREEBODY NUMBER IS THE BIT NUMBER.
FOR NODES FREEBODY NUMBER IS BIT NUMBER MINUS 15.
C
DO 220 K1 = 1, NFREE
IBITS = K1 + 15
DO 210 ELEMM = 1, UELEM
  CALL ELINFO (ELUM, ITEPE, NUEL, IFLAG, ISTART, LINES)
IF (BTEST(IFBODY(ELNUM),K1) .AND. ITYPE.GT.0) THEN
  DO 200 K3 = 1, HNEL
    NODE = NOP(K3,ELNUM)
    IFBODY(NODE) = IBSET(IFBODY(NODE),IBITS)
  CONTINUE
ENDIF
210 CONTINUE
220 CONTINUE
C
CALL ERRORS (0, 2, 'REVIEW')
RETURN
END
C
C  ====================== UTILIT =======
C
C INCLUDE (PROCESS)
C SUBROUTINE UTILIT
IMPLICIT REALS (A-H.O-Z)
C .  .SWITCHES: RENUMB=100:10,FORHAT=900:10
C . .SWITCHES:
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C NO SUBROUTINES OR FUNCTIONS CALLED FROM THIS PROGRAM UNIT
C
C ****************************************************************************
C
C THE ENTRIES IN THIS SUBROUTINE ARE USED TO SWAP, STORE AND RECOVER
C INFORMATION USED IN THE NONLINEAR PROCEDURE.
C
C ENTRY SWAP
C
ISWAP = LDEV1
LDEV1 = LDEV2
LDEV2 = ISWAP
C
RETURN
C
ENTRY SWAP1
C
LDEV1 = LDEV2
LDEV2 = ISWAP
LDEV3 = LDEV4
LDEV4 = ISWAP
C
C
RETURN
ENTRY SWAP2
ENTRY SWAP2
LDEV1 = LDEV2
LDEV2 = LDKEEP
RETURN
ENTRY SWAP3
ENTRY SWAP3
ISWAP = LDEV1
LDEV1 = LDEV3
LDEV3 = ISWAP
RETURN
ENTRY REWIN
ENTRY REWIN
REWIND (LDEV1, ERR=190, IOSTAT=IERROR)
REWIND (LDEV2, ERR=190, IOSTAT=IERROR)
REWIND (LDEV3, ERR=190, IOSTAT=IERROR)
RETURN
ENTRY RESTOR
ENTRY RESTOR (MDF, ISTART, NTDF, IDOF)
READ (LDEVST, *) ISTART, NTDF, NWMAX, LDEV1, LDEV2, LDEV3
DO 100 K1 = 1, MDF
READ (LDEVST, *) RE(K1), UTOTAL(K1)
100 CONTINUE
REWIND (LDEVST, ERR=190, IOSTAT=IERROR)
RETURN
ENTRY STORE
ENTRY STORE (FRACT, MDF, INCREM, NTDF, IDOF)
WRITE (LDEVST, *) FRACT, INCREM, NTDF, NWMAX, LDEV1, LDEV2, LDEV3
DO 110 K1 = 1, MDF
WRITE (LDEVST, *) RE(K1), UTOTAL(K1), IDOF(K1)
110 CONTINUE
REWIND (LDEVST, ERR=190, IOSTAT=IERROR)
WRITE (LDEVST, *) INCREM, PRESS, UTOTAL(2), UTOTAL(2114), PLWORK
PLWORK = 0
RETURN
ENTRY STORE1
ENTRY STORE1 (MDF, NTDF, IDOF)
WRITE (15, *) NTDF, LDEV1, LDEV2, LDEV3
DO 120 K1 = 1, MDF
WRITE (15, *) RE(K1), UTOTAL(K1)
120 CONTINUE
REWIND (16, ERR=190, IOSTAT=IERROR)
RETURN

ENTRY RESTR1

ENTRY RESTR1 (MDF, BTDF, IDOF)
READ (16, *) BTDF, LDEV1, LDEV2, LDEV3
DO 130 K1 = 1, MDF
READ (15, *) RE(K1), UTOTAL(K1)
130 CONTINUE
REWIND (16, ERR=190, IOSTAT=IERROR)
RETURN

ENTRY IOGET

ENTRY IOGET (IDEV, LENGTH, FMAT, N)
READ (IDEV, FMT=FMAT(1:N)) BUFFER(1:LENGTH)
RETURN

ENTRY IOPUT

ENTRY IOPUT (IDEV, LENGTH, FMAT, N)
WRITE (IDEV, FMT=FMAT(1:N)) BUFFER(1:LENGTH)
RETURN

ENTRY IGBKS

ENTRY IGBKS (IDEV)
BACKSPACE (UNIT=IDEV)
RETURN

ENTRY ARCHIV

ENTRY ARCHIV (MDF)
RETURN

ENTRY RECOV

ENTRY RECOV (FRACT, MDF, ISTART, BTDF, IDOF)
READ (LDEVST, *) FRACT, ISTART, BTDF, NWMAX, LDEV1, LDEV2, LDEV3
DO 140 K1 = 1, MDF
READ (LDEVST, *) RE(K1), UTOTAL(K1), IDOF(K1)
140 CONTINUE
REWIND (LDEVST, ERR=190, IOSTAT=IERROR)
RETURN

ENTRY WCONST

ENTRY WCONST
DO 150 K1 = 1, 40000
WRITE (4, *) COEFHP(K1), MPCDOF(K1)
150 CONTINUE
REWIND (4, ERR=190, IOSTAT=IERROR)
DO 160 K1 = 1, 6000
WRITE (15, *) EMPC, MPCMNT, MAXMPC, MPCADR(1,K1), MPCADR(2,K1)
160 CONTINUE
REWIND (15, ERR=190, IOSTAT=IERROR)
RETURN

ENTRY RCONST

ENTRY RCONST
ENTRY RCOHST

IF (IFLAG .EQ. 0) RETURN

READ (4, *) NPMP, MPCMST, LAMPC
DO 170 K1 = 1, 40000
READ (4, ") CEHMP(K1), MPDCDF(K1)
170 CONTINUE
REWIND (4, ERR=190, IOSTAT=IERROR)
DO 180 K1 = 1, 6000
READ (15, *) NPCP, MPCM, LAMPC, MPCLDR(K1,1), MPCLDR(K1,2)
180 CONTINUE
REWIND (15, ERR=190, IOSTAT=IERROR)
RETURN

STOP

FORMAT(180,1X,'ERROR IN REWINDING UTILITY FILES IS DETECTED',
1 ' BY ROUTINE CETAL1')
END

SUBROUTINE ERRORS (HERR, NEXT, MODULE)

C...SWITCHES: RENUM=100:10,FORMAT=900:10
C...SWITCHES:

INTEGER FMAT
CHARACTER*6 COMM, MODULE, CODE
CHARACTER*80 BUFFER, BUFF
CHARACTER*70 TEXT
COMMON/CDMP2/COMM, BUFFER, BUFF
DATA TERROR, IOUT /0, 13/
SAVE IERROR, IOUT

C THIS SUBROUTINE IS SETUP TO REPORT ERRORS ENCOUNTERED DURING
C INITIAL INPUT PROCESSING. NEED TO SET UP A FILE (UNIT 18) CONTAINING
C ERROR MESSAGES.

IF (NEXT .NE. 2) THEN
  IERROR = 1
  READ (18, REC=HERR) CODE, TEXT
  WRITE (IOUT, 900) CODE, MODULE, TEXT, BUFFER
  IF (NEXT .EQ. 0) THEN
    RETURN
  ELSE IF (NEXT .EQ. 1) THEN
    WRITE (IOUT, 910)
  ENDIF
ELSE IF (IERROR .EQ. 1) THEN
  WRITE (IOUT, 910)
  STOP
ENDIF

END
C INCLUDE(PROCESS)
SUBROUTINE IOCOPY
IMPLICIT REAL*8(A-H,O-Z)
C...SWITCHES: REUMB=100:10,FORMAT=900:10
C...SWITCHES:
C*******************************************************************************
C
C SUBROUTINES AND FUNCTIONS CALLED FROM THIS ROUTINE
C
C REWIN  SWAP
C
C*******************************************************************************

CHARACTER*240 DUMMY
CHARACTER*6 FMAT
COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDEKEEP,LDEVS,LDEVST
CALL REWIN
C
FMAT = ' (A96)'
100 CONTINUE
READ (LDEV1, FMT=FMAT(1:6), END=110) DUMMY(1:96)
WRITE (LDEV2, FMT=FMAT(1:5)) DUMMY(1:96)
GO TO 100
C
C
110 CONTINUE
CALL SWAP
C
RETURN
END
Vita

Ananth Ramaswamy was born on 11th January 1963, in Bombay, India. He graduated from the Indian Institute of Technology Madras, India, in July 1985, with a Bachelor of Technology degree in Civil Engineering. In December of 1986 he completed his post graduate work at the University of California at Davis, earning a Master of Science degree in Engineering.

He joined the graduate program at Louisiana State University in August 1987. He is currently a candidate for the Ph.D degree in Civil Engineering at Louisiana State University, Baton Rouge.
DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Ananth Ramaswamy

Major Field: Civil Engineering

Title of Dissertation: Nonlinear Inelastic Finite Element Analysis of Reinforced Concrete Structures with Emphasis on Shear and Torsion

Approved:

George F. Vassilatos
Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination: 3/13/92