1969

Selected Problems in the Design and Implementation of Direct Digital Control.

Charles Fred Moore

Louisiana State University and Agricultural & Mechanical College

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

Recommended Citation


https://digitalcommons.lsu.edu/gradschool_disstheses/1558
MOORE, Charles Fred, 1943-
SELECTED PROBLEMS IN THE DESIGN AND IMPLEMENTATION OF DIRECT DIGITAL CONTROL.

Louisiana State University and Agricultural and Mechanical College, Ph.D., 1969
Engineering, chemical

University Microfilms, Inc., Ann Arbor, Michigan
SELECTED PROBLEMS IN THE DESIGN AND IMPLEMENTATION
OF DIRECT DIGITAL CONTROL

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 Chemical Engineering

by

Charles Fred Moore
B.S., Louisiana State University, 1963
M.S., Louisiana State University, 1968
January, 1969
ACKNOWLEDGEMENT

This work was conducted under the guidance of Dr. Paul W. Murrill, Professor and Head, Department of Chemical Engineering at Louisiana State University and Dr. Cecil L. Smith, Technical Director Project THEMIS and Associate Professor of Chemical Engineering at Louisiana State University. The author is highly appreciative of their consideration, encouragement, and interest shown throughout this project. A special thanks is extended to Dr. Smith for the many hours of personal consultation and direction freely given from his busy daily schedule.

Thanks are also extended to other members of the Chemical Engineering Department for their assistance, as well as to Dr. Paul Julich of the Electrical Engineering Department. Much appreciation is extended to the personnel of the Computer Research Center, under the direction of Dr. B. B. Townsend.

The author wishes to express his appreciation for the support of this work by Project THEMIS Contract No. F44620-68-C-0021 administered for the U. S. Department of Defense by the U. S. Air Force Office of Scientific Research. Thanks are also extended to Dow Chemical Company whose fellowship supported the author for the past two years.

Sincere thanks are also extended to Mrs. Ruth Albright for her assistance with the preparation of this manuscript. The author also wishes to thank the Dr. Charles E. Coates Memorial Fund, donated by George H. Coates, for financial assistance in the preparation of this manuscript.

A special word of appreciation is given to the author's wife, Kitty, for her confidence and unending encouragement.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>viii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>ix</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>xv</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>I  INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>LITERATURE CITED</td>
<td>5</td>
</tr>
<tr>
<td>II EFFECT OF CONTROL ELEMENT/SENSOR DYNAMICS ON SYSTEM PERFORMANCE</td>
<td>6</td>
</tr>
<tr>
<td>Introduction</td>
<td>6</td>
</tr>
<tr>
<td>Hardware Lags in the Control Loop</td>
<td>6</td>
</tr>
<tr>
<td>Typical Control Loop</td>
<td>7</td>
</tr>
<tr>
<td>Mathematical Description</td>
<td>9</td>
</tr>
<tr>
<td>Index of Performance</td>
<td>11</td>
</tr>
<tr>
<td>Optimum Controller Parameters</td>
<td>13</td>
</tr>
<tr>
<td>Digital Simulation</td>
<td>15</td>
</tr>
<tr>
<td>System Response to a Step Disturbance</td>
<td>16</td>
</tr>
<tr>
<td>Quantitative Study</td>
<td>17</td>
</tr>
<tr>
<td>Error Ratio Method</td>
<td>19</td>
</tr>
<tr>
<td>Percent Increase in Error Method</td>
<td>26</td>
</tr>
<tr>
<td>Use of the Percent Increase in Error Plots</td>
<td>26</td>
</tr>
<tr>
<td>Practical Examples</td>
<td>29</td>
</tr>
<tr>
<td>Example 1 - Effect of Transmission Line Lag on a Reactor Under Direct Digital Control</td>
<td>29</td>
</tr>
<tr>
<td>Example 2 - Effect of Valve Speed on Composition Control Via Chromatograph</td>
<td>34</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>III</td>
<td>SIMPLIFYING DIGITAL CONTROL DYNAMICS FOR CONTROLLER TUNING AND HARDWARE LAG EFFECTS</td>
</tr>
<tr>
<td></td>
<td>Introduction</td>
</tr>
<tr>
<td></td>
<td>Development of Approximation</td>
</tr>
<tr>
<td></td>
<td>Applications and Examples</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
</tr>
<tr>
<td></td>
<td>LITERATURE CITED</td>
</tr>
<tr>
<td>IV</td>
<td>CONTROL OF A HIGH ORDER PLANT USING A TIME-OPTIMAL SECOND ORDER SWITCHING CURVE</td>
</tr>
<tr>
<td></td>
<td>Introduction</td>
</tr>
<tr>
<td></td>
<td>Complex Solution</td>
</tr>
<tr>
<td></td>
<td>Unmeasurable State Variable</td>
</tr>
<tr>
<td></td>
<td>Model Errors and Simplications</td>
</tr>
<tr>
<td></td>
<td>Unknown Equilibrium</td>
</tr>
<tr>
<td></td>
<td>Second-Order Feedback</td>
</tr>
<tr>
<td></td>
<td>Second-Order Approximation</td>
</tr>
<tr>
<td></td>
<td>Second-Order Plus Dead Time Approximation</td>
</tr>
<tr>
<td></td>
<td>Summary</td>
</tr>
<tr>
<td></td>
<td>LITERATURE CITED</td>
</tr>
<tr>
<td>V</td>
<td>FORMULATING THE NONLINEAR LEAST-SQUARE MODEL REGRESSION FOR FAST ON-LINE ANALYSIS</td>
</tr>
<tr>
<td></td>
<td>Introduction</td>
</tr>
<tr>
<td></td>
<td>&quot;Least-Square&quot; Regression</td>
</tr>
<tr>
<td></td>
<td>Development of a Nonlinear &quot;Least-Square&quot; Regression</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Data Collection</td>
<td>91</td>
</tr>
<tr>
<td>On-Line Regression</td>
<td>94</td>
</tr>
<tr>
<td>General On-Line Approach</td>
<td>98</td>
</tr>
<tr>
<td>On-Line Regression with Filtering</td>
<td>104</td>
</tr>
<tr>
<td>Comparison of Convention and On-Line Regression</td>
<td>106</td>
</tr>
<tr>
<td>Summary</td>
<td>106</td>
</tr>
<tr>
<td>NOMENCLATURE</td>
<td>108</td>
</tr>
<tr>
<td>LITERATURE CITED</td>
<td>109</td>
</tr>
<tr>
<td>VI FORMULATING THE LEAST-SQUARE REGRESSION FOR CONTINUOUS ANALYSIS</td>
<td>111</td>
</tr>
<tr>
<td>Introduction</td>
<td>111</td>
</tr>
<tr>
<td>Developing General Nonlinear Regression</td>
<td>112</td>
</tr>
<tr>
<td>Reducing Order of Search</td>
<td>114</td>
</tr>
<tr>
<td>Program Requirements Due to Summation Terms</td>
<td>118</td>
</tr>
<tr>
<td>Eliminating Summation Terms</td>
<td>119</td>
</tr>
<tr>
<td>Advantage of Weighted Average Formulation</td>
<td>121</td>
</tr>
<tr>
<td>Performance of the Weighted Average Formulation</td>
<td>123</td>
</tr>
<tr>
<td>Example - Non-Ideal Plug Flow Reactor</td>
<td>129</td>
</tr>
<tr>
<td>Summary</td>
<td>133</td>
</tr>
<tr>
<td>NOMENCLATURE</td>
<td>134</td>
</tr>
<tr>
<td>FLOW DIAGRAM KEY</td>
<td>135</td>
</tr>
<tr>
<td>LITERATURE CITED</td>
<td>136</td>
</tr>
<tr>
<td>VII PREDICTOR ALGORITHMS FOR DIRECT DIGITAL CONTROL</td>
<td>137</td>
</tr>
<tr>
<td>Introduction</td>
<td>137</td>
</tr>
<tr>
<td>Paradox of Digital Computer Control</td>
<td>137</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>2.1</td>
<td>Comparing Actual Sampled Data Results with Results of Continuous Dead Time Approximation</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Typical Process and Control System with Hardware Lag Before the Process</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Effect of Dimensionless Hardware Lag on System Response for a Continuous System Tuned by IAE</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Error Ratio Versus Dimensionless Hardware Lag for Continuous System Tuned by IAE</td>
<td>20</td>
</tr>
<tr>
<td>2.4a</td>
<td>Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for Continuous System Tuned by ISE</td>
<td>22</td>
</tr>
<tr>
<td>2.4b</td>
<td>Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for Continuous System Tuned by ITAE</td>
<td>23</td>
</tr>
<tr>
<td>2.4c</td>
<td>Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for Continuous System Tuned by IAE</td>
<td>24</td>
</tr>
<tr>
<td>2.5</td>
<td>Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of .40 x τ₁, tuned by IAE</td>
<td>25</td>
</tr>
<tr>
<td>2.6</td>
<td>Actual and &quot;Average&quot; Output of Zero-Order Hold</td>
<td>28</td>
</tr>
<tr>
<td>2.7a</td>
<td>Example 1 - Physical Description</td>
<td>32</td>
</tr>
<tr>
<td>2.7b</td>
<td>Example 1 - Mathematical Description</td>
<td>32</td>
</tr>
<tr>
<td>2.8a</td>
<td>Example 2 - Physical Description</td>
<td>35</td>
</tr>
<tr>
<td>2.8b</td>
<td>Example 2 - Mathematical Description</td>
<td>35</td>
</tr>
<tr>
<td>3.1a</td>
<td>Block Diagram of a Typical Digital Control Loop with a Sample and Zero Order Hold</td>
<td>40</td>
</tr>
<tr>
<td>3.1b</td>
<td>Block Diagram of an Equivalent Continuous Control Loop in Which G_d(s) Approximates the Behavior of the Digital Interface</td>
<td>40</td>
</tr>
<tr>
<td>3.2</td>
<td>&quot;True&quot; and &quot;Average&quot; Response of a Zero Order Hold to an Arbitrary Input</td>
<td>42</td>
</tr>
<tr>
<td>3.3</td>
<td>Comparison of Open Loop Response of a Discrete System with a Sample and Zero Order Hold to the Continuous System Using the Dead Time Approximation</td>
<td>46</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.4</td>
<td>Comparison of Closed Loop Response of a Discrete System with a Sample and Zero Order Hold to the Continuous System Using a Dead Time Approximation</td>
<td>47</td>
</tr>
<tr>
<td>3.5</td>
<td>Comparison of Frequency Response of a Discrete System with a Sample and Zero Order Hold to the Continuous System Using a Dead Time Approximation</td>
<td>48</td>
</tr>
<tr>
<td>3.6</td>
<td>Comparison of Optimum Proportional Controller Gains of a Discrete System with Gains Predicted by Dead Time Approximation</td>
<td>53</td>
</tr>
<tr>
<td>3.7a</td>
<td>Comparison of Optimum PI Controller Gains of a Discrete System with Gains Predicted by Dead Time Approximation</td>
<td>54</td>
</tr>
<tr>
<td>3.7b</td>
<td>Comparison of Optimum PI Controller Reset of a Discrete System with Reset Predicted by Dead Time Approximation</td>
<td>55</td>
</tr>
<tr>
<td>3.8</td>
<td>Comparison of True Percent Increase in Error Due to Increased Hardware Lag at Sampling Time = 0.2 to the Percent Increase in Error Predicted by the Dead Time Approximation</td>
<td>57</td>
</tr>
<tr>
<td>3.9</td>
<td>Comparison of Ultimate Gain of a Discrete System with Ultimate Gain Predicted by Dead Time Approximation</td>
<td>58</td>
</tr>
<tr>
<td>4.1</td>
<td>Time Optimal Controller for a Second Order Plant</td>
<td>66</td>
</tr>
<tr>
<td>4.2</td>
<td>Second-Order Model - Minimum Error Squared Fit</td>
<td>70</td>
</tr>
<tr>
<td>4.3</td>
<td>Time Response: Time Optimal Control of a 6th Order Plant Using a Second Order Model</td>
<td>71</td>
</tr>
<tr>
<td>4.4</td>
<td>Phase Plot: Time Optimal Control of a 6th Order Plant Using a 2nd Order Model</td>
<td>72</td>
</tr>
<tr>
<td>4.5</td>
<td>Second Order Plus Dead Time Model - Minimum Error - Squared Fit</td>
<td>73</td>
</tr>
<tr>
<td>4.6a</td>
<td>Compensating for Dead Time Using a Smith Predictor</td>
<td>75</td>
</tr>
<tr>
<td>4.6b</td>
<td>Equivalent Block Diagram Using a Perfect Smith Predictor (G_m(s) = G_p(s); T_m = T_s).</td>
<td>75</td>
</tr>
<tr>
<td>4.7</td>
<td>Smith Predictor with Time Optimal Controller</td>
<td>77</td>
</tr>
<tr>
<td>4.8</td>
<td>Time Response: Time Optimal Control of a 6th-Order Plant Using a 2nd Order Plus Dead-Time Model</td>
<td>78</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>4.9</td>
<td>Phase Plot of Predictor Output: Time Optimal Control of a 6th Order Plant Using a 2nd Order Plus Dead Time Model</td>
<td>79</td>
</tr>
<tr>
<td>4.10</td>
<td>Second Order Plus Dead Time Model - Minimum Error Squared Fit</td>
<td>80</td>
</tr>
<tr>
<td>4.11</td>
<td>Time Response: Time Optimal Control of a 6th Order Plant Using a 2nd Order Plus Dead Time Model</td>
<td>81</td>
</tr>
<tr>
<td>4.12</td>
<td>Phase Plot of Predictor Output: Time Optimal Control of a 6th Order Plant Using a 2nd Order Plus Dead Time Model</td>
<td>82</td>
</tr>
<tr>
<td>5.1</td>
<td>Micro-Flow Chart for On-Line Data Collection</td>
<td>92</td>
</tr>
<tr>
<td>5.2</td>
<td>Micro-Flow Diagram: On-Line Identification</td>
<td>100</td>
</tr>
<tr>
<td>6.1</td>
<td>Simplified Flow Diagram of &quot;Weighted-Average&quot; On-Line Least-Square Analysis of Process Data for a First-Order Lag Plus Dead Time Model</td>
<td>122</td>
</tr>
<tr>
<td>6.2</td>
<td>Performance of the &quot;Continuous&quot; Least-Square Regression to a Closed Loop Change in Set-Point ($G_p(s) = e^{-0.5s}/(1+s)$)</td>
<td>124</td>
</tr>
<tr>
<td>6.3</td>
<td>Performance of &quot;Continuous&quot; Least-Square Regression to Pure Noise ($G_p(s) = e^{-0.5s}/(1+s)$)</td>
<td>125</td>
</tr>
<tr>
<td>6.4a</td>
<td>Dynamic Response of the &quot;Continuous&quot; Least-Square Regression to a Step Change in Process Time Constant</td>
<td>126</td>
</tr>
<tr>
<td>6.4b</td>
<td>Dynamic Response of the &quot;Continuous&quot; Least-Square Regression to a Step Change in Process Gain</td>
<td>127</td>
</tr>
<tr>
<td>6.4c</td>
<td>Dynamic Response of the &quot;Continuous&quot; Least-Square Regression to a Step Change in Process Dead Time</td>
<td>128</td>
</tr>
<tr>
<td>6.5</td>
<td>Non-Ideal Plug Flow Reactor Under Computer Control</td>
<td>131</td>
</tr>
<tr>
<td>6.6</td>
<td>Model Profile at Various Operating Levels in a Non-Ideal Plug Flow Reactor Using Continuous Least-Square Analysis</td>
<td>132</td>
</tr>
<tr>
<td>7.1</td>
<td>Effect of Sampling Time on System Performance</td>
<td>138</td>
</tr>
<tr>
<td>7.2</td>
<td>Compensating for Digital Interface Using Analytical Predictor Scheme For Eliminating Dead Time</td>
<td>141</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>7.3</td>
<td>Effect of Dead Time on Sampled Process Information</td>
<td>144</td>
</tr>
<tr>
<td>7.4a</td>
<td>Performance of the Predictor Algorithm to a Change in Set-Point -- Perfect Modeling</td>
<td>151</td>
</tr>
<tr>
<td>7.4b</td>
<td>Performance of the Predictor Algorithm to a Change in Load -- Perfect Modeling</td>
<td>151</td>
</tr>
<tr>
<td>7.5</td>
<td>Effect of Hard Constraints on the Predictor-Algorithm</td>
<td>155</td>
</tr>
<tr>
<td>7.6a</td>
<td>Effect of Severe Modeling Errors on the Predictor-Algorithm -- Set-Point Change</td>
<td>157</td>
</tr>
<tr>
<td>7.6b</td>
<td>Effect of Severe Modeling Errors on the Predictor-Algorithm -- Load Change</td>
<td>158</td>
</tr>
<tr>
<td>7.7</td>
<td>Computer Control of a Non-Ideal Plug Flow Reactor</td>
<td>161</td>
</tr>
<tr>
<td>7.8a</td>
<td>A Set-Point Change</td>
<td>162</td>
</tr>
<tr>
<td>7.8b</td>
<td>Control of a Non-Ideal Plug Flow Reactor Using P-A and PI Algorithms</td>
<td>162</td>
</tr>
<tr>
<td>A.1</td>
<td>Macro Flow Chart - Multidimensional Optimization Using Subroutine PATERN</td>
<td>172</td>
</tr>
<tr>
<td>B.1</td>
<td>Effect of Dimensionless Hardware Lag on System Response for a Continuous System Tuned by ISE</td>
<td>183</td>
</tr>
<tr>
<td>B.2</td>
<td>Effect of Dimensionless Hardware Lag on System Response for a Continuous System Tuned by ITAE</td>
<td>184</td>
</tr>
<tr>
<td>B.3</td>
<td>Effect of Dimensionless Hardware Lag on System Response for a Continuous System Tuned by IAE</td>
<td>185</td>
</tr>
<tr>
<td>B.4</td>
<td>Effect of Dimensionless Hardware Lag on System Response for a Sampling Time of 0.10 x τ₁, Tuned by IAE</td>
<td>186</td>
</tr>
<tr>
<td>B.5</td>
<td>Effect of Dimensionless Hardware Lag on System Performance for a Sampling Time of 0.20 x τ₁, Tuned by IAE</td>
<td>187</td>
</tr>
<tr>
<td>B.6</td>
<td>Effect of Dimensionless Hardware Lag on System Performance for a Sampling Time of 0.30 x τ₁, Tuned by IAE</td>
<td>188</td>
</tr>
<tr>
<td>B.7</td>
<td>Effect of Dimensionless Hardware Lag on System Response for a Sampling Time of 0.40 x τ₁, Tuned by IAE</td>
<td>189</td>
</tr>
</tbody>
</table>
Effect of Dimensionless Hardware Lag on System Response for a Sampling Time of 0.50 x $\tau_1$, Tuned by IAE

Error Ratio Versus Dimensionless Hardware Lag for a Continuous System Tuned by ISE

Error Ratio Versus Dimensionless Hardware Lag for a Continuous System Tuned by ITAE

Error Ratio Versus Dimensionless Hardware Lag for a Continuous System Tuned by IAE

Error Ratio Versus Dimensionless Hardware Lag for a Sampling Time of 0.10 x $\tau_1$, Tuned by IAE

Error Ratio Versus Dimensionless Hardware Lag for a Sampling Time of 0.20 x $\tau_1$, Tuned by IAE

Error Ratio Versus Dimensionless Hardware Lag for a Sampling Time of 0.30 x $\tau_1$, Tuned by IAE

Error Ratio Versus Dimensionless Hardware Lag for a Sampling Time of 0.40 x $\tau_1$, Tuned by IAE

Dimensionless Hardware Lag versus Dimensionless Dead Time at Constant Increase in Error for a Continuous System Tuned by ISE

Dimensionless Hardware Lag versus Dimensionless Dead Time at Constant Increase in Error for a Continuous System Tuned by ITAE

Dimensionless Hardware Lag versus Dimensionless Dead Time at Constant Increase in Error for a Continuous System Tuned by IAE

Dimensionless Hardware Lag versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of 0.10 x $\tau_1$, Tuned by IAE

Dimensionless Hardware Lag versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of 0.20 x $\tau_1$, Tuned by IAE

Dimensionless Hardware Lag versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of 0.30 x $\tau_1$, Tuned by IAE
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.23</td>
<td>Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of $0.40 \times \tau_1$, Tuned by IAE</td>
<td>205</td>
</tr>
<tr>
<td>B.24</td>
<td>Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of $0.50 \times \tau_1$, Tuned by IAE</td>
<td>206</td>
</tr>
</tbody>
</table>
ABSTRACT

In this dissertation several specific problems are considered in the design and implementation of digital computer control applied to chemical processes with emphasis placed on economics and the efficient utilization of both computer time and storage. No experimental data were taken and the entire study was done using digital simulation on an IBM 7040.

First, generalized graphs are presented from which the dynamic contribution of a control loop hardware element can be gauged and hopefully used in the economic selection and/or design of control loop hardware.

Second, it is demonstrated that in most situations the dynamic effect of a sampler and zero order hold is similar to the dynamic effect of a pure dead time of one half the sampling time. For processes already containing a dead time term, this results in a significant simplification in the design and/or selection of control loop hardware and in the implementation of controller tuning.

Third, a practical method to implement time optimal control of a high order process is demonstrated. The method consists of characterizing the high order process as a second order lag plus dead time and implementing the resulting second order time optimal switching curve along with a Smith predictor to eliminate the dead time.

Fourth, a method is presented for formulating the conventional non-linear least-square regression which results in sufficient reduction in both computer time and storage such that true "on-line" analysis is possible.
Fifth, a method is shown by which the multidimensional search used by most nonlinear least-square regression programs can be reduced in many cases to a one-dimensional search. Also, shown is a modification of the least square regression such that the summation of the cross product and product terms are replaced by a weighted average estimate of the values. The result is a fast efficient program which can be actually used to continuously analyse either closed loop or noise data.

Finally, a predictor algorithm is developed specifically for the control of a first order lag plus dead time under direct digital control. The result is an algorithm which performs much better than the conventional one-, two-, and three-mode controllers even under such non-ideal conditions as constraints on the allowable control action and severe modeling errors.
CHAPTER I

INTRODUCTION

In recent years the digital computer has had a real impact in the design and implementation of process control systems in the chemical process industry. Improved computer technology including analog-to-digital interface now make it possible to directly use the digital computer to control chemical processes. The memory, logic capabilities and general flexibility of the digital computer over the conventional analog controller offers the control engineer much more power and freedom in the design and implementation of process control.

This dissertation will examine and study several aspects of the direct control of chemical processes using digital computation. Fundamental in each study is the recognition that along with the tremendous versatility and power of the digital controller comes a considerable increase in expenditure forcing the control engineer to be primarily concerned about using the potential of the computer effectively and efficiently.

In the second chapter the problem of the economic selection of accompanying hardware elements, such as control valves and sensing devices, on the basis of dynamics as well as capital cost, is considered, recognizing the fact that the performance of the control system is limited by the dynamics of both the process and the control loop hardware. In some cases the system performance can be significantly improved by simply spending the money necessary to install a "fast" hardware element. In other cases, regardless of the attention
and money spent on the element, no significant improvement can be made in system performance.

In order to gauge the effect of hardware dynamics on system performance a control loop consisting of a proportional plus integral controller, a first order lag plus dead time process, and a hardware element described as a pure first order lag was studied for both continuous and sampled data control. The results of the study are presented in several plots which can be used to give a fairly accurate estimate of the economic "cost" of the hardware in terms of system performance.

In Chapter III the fundamental difference between the conventional analog loop and the sampled data loop associated with the digital controller is investigated. The behavior of the digital interface is shown to be very similar in nature to the behavior of a pure dead time of one sampling time. This approximation is shown to greatly simplify the hardware sizing procedure for computer control as outlined in Chapter II and is also effective in reducing the complexity of tuning digital algorithms by enabling the use of various tuning equations developed for continuous control.

In Chapter IV the time optimal control of the typical high order overdamped systems encountered in the chemical process industries is considered. The theoretical time optimal controller developed using the minimum principle is not practical to implement due to the uncertainties inherent in the control of chemical process and due to the tremendous computational requirements involved. In this chapter, however, a practical approach to the problem is demonstrated by the characterization of the process by a second order lag plus dead time
and the implementation of a second-order feedback switching curve using the "Smith Predictor" (1) to remove the effective dead time from the process loop.

Chapters V and VI investigate the problem of on-line process model determination. Much of the modern control techniques, including the time-optimal controller in Chapter IV and the predictor-algorithm presented on Chapter VII, depend heavily on a fairly accurate mathematical description of the process under control. In order to utilize the full potential of the technique there is a strong need to identify and follow actual changes in process parameter. Conventional methods for determining model parameters are bulky for use in an on-line real time environment yet in the past they have been used even at the expense of both valuable computer time and storage.

Chapter V demonstrates how the conventional least-square model regression can be formulated to analyse pulse data on-line with considerable reduction in both computer time and computer storage over conventional formulations.

Chapter VI on the other hand demonstrates how for most common models used in the chemical process industries the multidimensional search characteristic of the conventional least-square technique can be reduced to a much faster one-dimensional search. It also demonstrates how the summation terms in the conventional least-square problem can be replaced by a "weighted-average" estimate of the product and cross product terms. This substitution results in the elimination of the long data tables and permits fast, continuous on-line analysis of closed loop data which can actually be used with good success for an input of pure noise.
In an attempt to break free of the conventional one-, two-, and three-mode controllers Chapter VII develops a general direct digital control algorithm designed specifically for the control of a first order lag plus dead time process. The results are a relatively simple algorithm which is a combination of a non-interacting proportional and reset feedback modes and a dynamic feedforward mode. The algorithm for both perfect modeling and for severe modeling errors perform much better than the conventional algorithms to both changes in set-point and to changes in load.

In summary the purpose of this dissertation is to focus on several aspects of the digital computer control and presents in these area practical approaches to the design and implementation of the digital computer in the control of chemical processes.
LITERATURE CITED

CHAPTER II

EFFECT OF CONTROL ELEMENT/SENSOR DYNAMICS ON SYSTEM PERFORMANCE

Introduction

With the advent of the high speed digital computer working as a controller much work has been done on control strategies and algorithms which were previously impractical or impossible to implement using conventional analog control techniques. Such concepts as parameter optimization, on-line tuning, optimal control, etc., have made it possible to effectively eliminate and/or optimize the dynamics associated with the controller and in some cases to greatly tighten the overall control loop. This has forced the control engineer to be increasingly more concerned about the dynamic contribution of the other hardware components in the control loop. Control engineers have also become much more concerned with optimizing the design of conventional analog control loops.

This chapter examines several aspects of hardware dynamics and presents quantitative results which can be applied in the selection and/or design of supporting hardware to more economically utilize the potential of the digital computer as a controller. The results also will provide insight to the more practical design of conventional analog control loops.

Hardware Lags in the Control Loop

The control valve, final manipulator, or final control element is
one of the first areas of concern when considering hardware dynamics. One question worthy of concern is the "speed" of this element -- hereafter referred to as a valve. An increasingly large range of valve dynamics are becoming available to the control engineer, ranging from the conventional pneumatic-diaphragm actuation to the sophisticated stepping motor and fast hydraulic positioners. However, along with the improved valve dynamics there is a corresponding increase in cost. Thus the question -- when can a significant improvement in system response be realized to justify the additional cost associated with the faster valve?

Similar concern arises over other hardware devices such as the sensor element in the feedback loop. Intuitively the lag associated with a thermocouple will contribute error to the overall control of the system, but how much error?

The general question becomes, "When do hardware lags in the control loop become critical to the operation of the system?"

Typical Control Loop

To better understand quantitatively the role of the control valve, sensing element, and other dynamic hardware devices, a typical control loop was studied. The system considered is shown in Figure 2.1. It consists of a proportional-plus-integral controller, a hardware element characterized by a pure first-order lag, and a process characterized by a pure first-order lag plus dead time. A sampler and a zero-order hold are also included for the sampled data case.

In reality the process will probably be much more complicated; however, the approximation of a large majority of the process typically
Figure 2.1 Typical Process and Control System with Hardware Lag Before the Process.
encountered in practice by a first order lag plus dead time is common
(1).

Mathematical Description

The response of the system for the continuous case can be charac-
terized by the following transfer function:

\[ C(s) = \frac{G_1(s)D(s)}{1 + G_c(s)G_1(s)G_2(s)} \]  \hspace{1cm} (2.1)

where \( s \) = dummy Laplace variable

\[ G_c(s) = K_o \left[ 1 + \frac{1}{\tau_c s} \right] \]  \hspace{1cm} (controller)

\[ G_1(s) = \frac{K_1 e^{-\theta_1 s}}{1 + \tau_1 s} \]  \hspace{1cm} (process)

\[ G_2(s) = \frac{K_2}{1 + \tau_2 s} \]  \hspace{1cm} (hardware element)

\[ D(s) = \frac{1}{s} \]  \hspace{1cm} (unit step disturbance)

The system parameters are defined as follows:

\( K_o \) = Controller gain

\( K_1 \) = Process gain

\( K_2 \) = Hardware element gain
\( K_c \) = Loop gain \((K_c = K_0 K_1 K_2)\)

\( T_1 \) = Reset time in controller

\( \tau_1 \) = Time constant of process

\( \tau_2 \) = Time constant of hardware element

\( \theta \) = Transport lag (time delay) of process

Likewise, for the sampled data case it is convenient to describe the system response by z-transforms.

\[
C(z) = \frac{G_1 D(z)}{1 + G_c(z)G_1 G_2 H(z)} \quad (2.2)
\]

where \( G_1 D(z) = Z[G_1(s)D(s)] \)

\( G_1 G_2 H[z] = Z[G_1(s)G_2(s)H(s)] \)

\[
G_c(z) = K_c \left[ 1 + \frac{T_s}{T_1} \left( \frac{z}{z-1} \right) \right] \quad \text{(PI Controller)}
\]

Throughout this article, the hold element will always be a zero-order hold, thereby making

\[
H(s) = \frac{1-e^{-sT_s}}{s}
\]

Without solving these two expressions it is possible to say in general that the time response, \( c(t) \), of the system is a function of the system parameters, the hardware parameters, the controller parameters, the sampling time, and the initial conditions. Setting all initial conditions to zero the response can be expressed mathematically as:
\[ c(t) = \theta (T_s, K_c, T_1, \tau_1, \tau_2, \Theta_0) \]  \hspace{1cm} (2.3)

where \( c(t) \) represents the entire time response from \( t = 0 \) to \( t = \infty \).

**Index of Performance**

In order for the study to be meaningful a criterion is needed to quantitatively determine when one response \( c_1(t) \) is "better" than another response \( c_2(t) \). Selecting such a criterion is a difficult task because it invariably falls back on the question of what is the "best" response. Recognizing the fact that there is no universal criterion for "good" response, one must be selected.

A sound basis on which to define a "good" response is economics. The best response is the response which yields the highest profit or which results in the lowest operating cost. Typically in most process operations it is desirable to maintain the system output at some fixed value. In reality upsets enter the system forcing the output away from the desired value. A control loop is usually included to take corrective action; however, as long as the desired output is off specification some economic penalty will be paid until the output returns to its desired value.

In many cases a convenient way to define a performance index which approximates the economics is by an integral criteria of the form:

\[ J(K_c, T_1, T_s, \tau_1, \tau_2, \Theta_0) = \int_0^\infty L(e(t), t) dt \]  \hspace{1cm} (2.4)

where \( L \) is some economic function of the output error and time. \( J \) is the economic "cost" associated with a particular response, \( c(t) \), and
is therefore a function of all the system parameters. (J is also a function of the disturbance, d(t); however, throughout this chapter only a unit step is considered.)

No doubt the unique economics of individual systems would dictate numerous different functions L, some of which could be dependent on system variables other than those which are required to specify e(t) (e.g. the manipulated variable to include the cost of fuel). Nevertheless, three criteria which are representative of typical system economics are |e(t)|, e²(t), and t|e(t)| (1). Note that |e(t)| assumes that the cost is a function only of the magnitude of the output deviation, whereas e²(t) penalizes large deviations much more severely than for small deviations. Similarly, t|e(t)| weighs the error with time to penalize severely for even small errors that persist at large values of time after the upset has entered the system.

The integral criteria corresponding to the three functions mentioned above were chosen for this study and are defined in the following manner:

\[
\text{ISE} = \int_0^\infty e^2(t) dt \\
\text{IAE} = \int_0^\infty |e(t)| dt \\
\text{ITAE} = \int_0^\infty t|e(t)| dt
\] (2.5) (2.6) (2.7)

Other such criteria could be defined, the only restriction being that they must be positive for all t and e(t). However, these three have
received most of the attention in the literature.

**Optimum Controller Parameters**

Before any valid response comparisons can be made for different hardware elements the controller must be well tuned. In general the optimum gain, \( K_c^* \), and reset time, \( T_i^* \), are both functions of the process, the hardware element, and the sampling time. That is,

\[
K_c^* = f_1(T_s, \tau_1, \tau_2, \theta_0)
\]  

\[
T_i^* = f_2(T_s, \tau_1, \tau_2, \theta_0)
\]

Therefore, for each case investigated \( K_c^* \) and \( T_i^* \) must be re-evaluated.

In the previous section "good" response was defined in terms of an index of performance, \( J \), which was representative of the economic cost paid for being off specification. The "optimum" control parameters, therefore, correspond to the minimum cost or the minimum of the selected index of performance. Mathematically at the minimum index of performance the following relationships hold:

\[
\frac{\partial}{\partial K_c} \left[ J(K_c, T_i, T_s, \tau_1, \tau_2, \theta_0) \right] = 0 \]

\[
J = J_{\text{min}}
\]

\[
\frac{\partial}{\partial T_i} \left[ J(K_c, T_i, T_s, \tau_1, \tau_2, \theta_0) \right] = 0 \]

\[
J = J_{\text{min}}
\]

where

\[
J_{\text{min}} = \min_{T_i, K_c} \int_0^\infty L(e(t), t) dt
\]

or in terms of the system parameters
Note that the index of performance, $J$, in general is a function of all the system variables but the optimum index of performance, $J^*$, is dependent only on the process parameters (including the sampling time) and is not a function of the controller parameters.

Assuming the above equations can be solved, the problem of comparing responses is simplified by combining either Equation (2.1) and (2.2) with Equations (2.8) and (2.9). This results in a new function describing the response, $c(t)$, which is independent of $K_c$ and $T_i$, namely,

$$c(t) = \theta(T_s, \tau_1, \tau_2, \theta_o)$$

The problem can be further simplified by writing the above function in terms of normalized or dimensionless variables. One convenient set of variables is obtained by dividing all the independent variables by $\tau_1$, giving:

$$\left(\frac{t}{\tau_1}\right) = \text{dimensionless time}$$
$$\left(\frac{T_s}{\tau_1}\right) = \text{dimensionless sampling time}$$
$$\left(\frac{\tau_2}{\tau_1}\right) = \text{dimensionless hardware time constant}$$
$$\left(\frac{\theta_o}{\tau_1}\right) = \text{dimensionless transport lag}$$
$$\left(\frac{T_i}{\tau_1}\right) = \text{dimensionless process time constant} = 1$$

Since the dimensionless process time constant is always unity, the system can be completely described by the following function:

$$c\left(\frac{t}{\tau_1}\right) = \theta_2\left[\frac{T_s}{\tau_1}, \left(\frac{\tau_2}{\tau_1}\right), \left(\frac{\theta_o}{\tau_1}\right)\right]$$  \hspace{1cm} (2.12)\]

Likewise, the optimum index of performance can be simplified and
expressed in the same manner.

\[ J_{\text{min}} = J^* \left[ \frac{T_s}{\tau_1}, \frac{\tau_2}{\tau_1}, \frac{\theta}{\tau_1} \right] \quad (2.13) \]

The problem is now in a convenient form to study the effect of the hardware time constant, \((\tau_2, \tau_1)\), for various combinations of \((T_s/\tau_1)\) and \((\theta/\tau_1)\) on both the system response \(c(t/\tau_1)\) and the minimum index of performance, \(J^*\).

**Digital Simulation**

A numerical method was selected to solve the necessary equations. The existence of the transport lag in the closed loop transfer function (Equations 2.1 and 2.2) and the nonlinear index of performance render analytical approaches impractical. Also, with the absence of explicit analytical solutions, determination of the optimum controller parameters becomes a two dimensional search for different values of \(K_c\) and \(T_i\) which minimize the index of performance \(J\). The logic involved to conduct such a search lends itself to the digital computer over other methods such as pure analog simulation.

The digital simulation of the control loop is actually a convenient way to approach the problem. The entire control loop, including the controller, the sampler and hold element, the process, and the hardware device, is easily simulated using finite difference approximation. In this manner \(c(t)\) can be generated and the integral criteria calculated for a unit step disturbance and a given set of controller and process parameters. Optimization of the controller parameters can be easily accomplished by using an optimization subroutine which, by a trial-and-error procedure, determines values of \(K_c\) and \(T_i\)
which minimize the selected index of performance.

In this study a pattern search strategy was used to determine the controller parameters (2) (see Appendix A). Selected points were checked using optimum gradient (3) to insure that the criterion was actually being minimized. All work was done on an IBM 7040 digital computer and the total study represents about 100 hours of machine time.

**System Response to a Step Disturbance**

To qualitatively determine the effect of the hardware element on system performance it is instructive to observe the actual response of a system to a step disturbance.

In this study it was interesting to note that except at high sampling times the general shape of the response for a fixed integral criteria remained about the same; however, each response was easily distinguishable by a difference in amplitude and period (see Figure 2.2). In general it appears that the amplitude and period increase almost directly with an increase in hardware lag and/or an increase in sampling time. To study this effect a process of $\theta_o/\tau_1 = 0.20$ was investigated for the continuous case ($T_s/\tau_1 = 0$) and for the sampled data case at various sampling times ($T_s/\tau_1$) between 0.1 and 0.5 the system response was obtained for values of hardware lag ranging from $\tau_2/\tau_1 = 0$ to $\tau_2/\tau_1 = 1.0$.

Figure 2.2 is an example of one such plot obtained in this study. It depicts the effect of various values of $\tau_2/\tau_1$ for the case when the process ($\theta_o/\tau_1 = 0.20$) is continuously controlled. For each $\tau_2/\tau_1$ the controller parameters were chosen (tuned) to minimize IAE. Other plots obtained in this study can be found in detail in Appendix B. At large sampling times and low hardware lags the expected shape of the response
changes somewhat since at very large sampling times the system becomes "sampling time limited". In effect the sampling interval is so large that the dynamics of the system tend to die out between each change in control action, resulting in a rough, jerky response curve.

In the continuous case the plots corresponding to Figure 2.2 also were obtained for ISE and ITAE in addition to IAE. Essentially the same effect due to $\tau_2/\tau_1$ and $T_s/\tau_1$ was observed for each criterion. However, the general shape of the system response was characteristically different for each criterion. As would be expected ISE yields the lowest initial overshoot of the three but by far the largest settling time. ITAE on the other hand, generally gave the most rapid settling time but by far the largest initial overshoot. IAE turned out to be a compromise between the two, yielding a settling time much shorter than ISE and an overshoot somewhat less than ITAE.

Quantitative Study

A knowledge of the general system response is most helpful in developing an "intuitive feel" for the effects of the hardware element; however, it would be desirable to approach the problem quantitatively, i.e., to be able to objectively say how much economic gain could be obtained from improved response by using one hardware element as opposed to another.

In an attempt to provide such an approach, the three integral criteria previously discussed were assumed to be an indication of economic cost. The various integral criteria were thus evaluated for numerous combinations of $T_s/\tau_1$, $\Theta_o/\tau_1$, and $\tau_2/\tau_1$.

Two methods, error ratio and percent increase in error, were chosen to illustrate the results obtained from this study. Both will
Figure 2.2. Effect of Dimensionless Hardware Lag on System Response for a Continuous System Tuned by IAE.
be described and discussed in detail in the following sections. Both methods contain the same information but provide two different viewpoints from which to study the results. For the sake of brevity only the continuous case with IAE integral criteria is illustrated by the Error Ratio plot (Figure 2.3) and only a select few of the Percent Increase in Error plots (Figures 2.4 and 2.5) are included. The actual study investigates both methods for a range of sampling times including $\frac{T_s}{\tau_1} = 0.0, 0.1, 0.2, 0.3, 0.4$ and $0.5$. In the continuous case all three integral criteria are investigated while the sampled data case includes only IAE. A complete set of plots for each method can be found in Appendix B.

**Error Ratio Method**

One convenient way to look at the results is, for a given process and sampling time, to define the error ratio to be the index of performance $J^*$ evaluated at $\tau_2/\tau_1$ divided by the index of performance evaluated at $\tau_2/\tau_1 = 0$ (no lag or perfect element). Mathematically:

$$\text{Error Ratio} = \frac{J^* \left[ (T_s/\tau_1), (\tau_2/\tau_1), (\theta_0/\tau_1) \right]}{J^* \left[ (T_s/\tau_1), (0.0), (\theta_0/\tau_1) \right]}$$

The error ratio can be plotted versus $\tau_2/\tau_1$ with $\theta_0/\tau_1$ as a parameter, as in Figure 2.3.

The following are several observations and conclusions which can be drawn from the study of the error ratio.

**Effect of Dead Time:** The most outstanding result which is obvious from Figure 2.3 is the large effect of the dead time on the sensitivity of the system to a hardware lag. At very large dead times the system is relatively insensitive to changes in $\tau_2/\tau_1$; however, for small $\theta_0/\tau_1$
Figure 2.3 Error Ratio Versus Dimensionless Hardware Lag for Continuous System Tuned by IAE.
the system becomes extremely sensitive. Note that in Figure 2.3 for a small $\Theta_0 / \tau_1$ a hardware lag of only one tenth the value of the process lag will increase the integral of the error squared by a factor of eight over the case where the hardware lag is negligible. This is the result of the effective increase in order of the system with the increase in dead time. At very low dead times the process (including the hardware lag) is essentially second order and much more sensitive to a change in position of one of the poles of the transfer function than a much higher order system (corresponding to high dead times).

Effect of Criterion Function: The three integral criteria differ in degree of sensitivity to the hardware lag. ITAE is by far the most sensitive for all values of dead time. IAE is the least sensitive of the three especially at low values of dead time. (This is a factor which might well be considered in selection of the proper performance criteria for a particular system.)

Effect of Sampling Time: The error ratio was also used to show the effect of the hardware lag at various sampling intervals. It was noted that at higher sampling times the sensitivity at lower dead times decreased considerably whereas at very high dead times it remained about the same. Again this can intuitively be explained by the effective increase in order due to an increase in dead time. The higher-order system is not as sensitive to sampling time as a simple, low-order system.

It was interesting to note in the study of the sampled data case at very large sampling times and low dead times the plots appear to be out of place (judging from the shape of the other plots at lower sampling rates). Actually what is demonstrated is the transition between
Figure 2.4a Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for Continuous System Tuned by ISE.
Figure 2.4-b. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for Continuous System Tuned by ITAE.
Figure 2.4-c. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for Continuous System Tuned by IAE.
Figure 2.5. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of $0.40 \times \tau_1$, tuned by IAE.
a system which is dynamics limited to a system which is sampling time limited. At high sampling times and low dead times the system is almost reaching a steady state value between sampling times.

**Percent Increase in Error Method**

Another and perhaps more meaningful way to consider the effect of the hardware lag is a series of plots of $\frac{\tau_2}{\tau_1}$ versus $\frac{\Theta_0}{\tau_1}$ for a fixed increase in performance index, defined,

\[
\text{\% Increase in Error} = (\text{Error Ratio}) \times 100
\]

The most instructive of the plots obtained in this manner are illustrated in Figures 2.4 and 2.5. Essentially the same general conclusions can be drawn from observing the percent increase in error as were discussed in the previous section. However, in the discrete sampling case an interesting phenomenon occurs which is not apparent when the results are observed in terms of the Error Ratio plots. As demonstrated in Figure 2.5 in the sampled data case the plots are not the smooth function as observed for the continuous case (Figure 2.4). Instead a curious break appears in each line. The curious part about the break is that it occurs in each line at precisely the dead-time equal to a discrete integer multiple of the sampling time. In general the effect was observed to be much more pronounced at higher sampling times and scarcely detectable at the lower more reasonable sampling times. In effect this means that the system, especially at large sampling times, is less sensitive to dynamics of the hardware lag when the dead time happens to be an integer multiple of the sampling time.

**Use of the Percent Increase in Error Plots**

The percent increase in error plots, illustrated in Figures 2.4
and 2.5 provide a convenient means by which to compare the effects of various hardware elements. For a given process \( (\theta_0 \text{ and } \tau_1) \) the deterioration of the system performance due to a particular hardware element \((\tau_2)\) can be determined directly by reading the percent increase in error from the appropriate plot (determined by the sampling time of the system and the desired index of performance). The percent increase in error approximates the incremental increase in "cost" of system operation with the hardware element \(\tau_2\) over the corresponding cost of operation if the hardware element were "perfect" in the sense of an instantaneous response.

The following section more clearly illustrates the use of these plots by several specific examples. However, before this discussion one interesting simplification should be examined.

Consider the response of the zero order hold shown by the solid line in Figure 2.6. If this output is averaged over each sampling time an average response can be fairly well represented by the dotted line connecting the midpoint of each holding period. In most process applications involving such a sample and hold device it is reasonable to expect the "plant" to be reasonably "slow" responding to short steps characteristic of the zero order hold. Therefore, it seems logical to expect the "average" signal to be descriptive of the hold in terms of the effect on the process output. Note, from Figure 2.6 the average signal can be described simply as the input signal to the sample and hold element delayed by one-half the sampling interval. If such intuitive logic is correct, the digital interface (the sample and hold element) can be replaced simply by a pure transport lag or dead time. It is interesting to note that for reasonable sampling times the results of
**Figure 2.6.** Actual and "Average" Output of Zero-Order Hold.
this study seems to support this simplification (see Table 2.1).

Even at fairly large sampling times \( T_0/\tau_1 = .3 \) and \( \tau_2/\tau_1 = .9 \) the approximation appears to be adequate except when the process dead time \( \theta_0/\tau_1 \) is less than 0.1 to 0.2. In general it appears that the approximation fails to describe the system adequately only for the cases in which the system becomes sampling time limited instead of dynamics limited.

This observation provides a convenient basis on which to simplify the use of the Percent Increase in Error plots. For most applications the continuous plots are all that is needed to evaluate a particular hardware element even for the case when the mode of control is actually discrete.

In order to use the continuous plots for the sampled data case the same procedure is used except an "equivalent" dead time \( \theta_0/\tau_1' \) is used instead of the actual dead time \( \theta_0/\tau_1 \), where the equivalent dead time is defined as follows:

\[
\left( \frac{\theta_0}{\tau_1} \right)' = \left( \frac{\theta_0}{\tau_1} \right) + \frac{1}{2} \left( \frac{T_0}{\tau_1} \right)
\]

Practical Examples

The following section contains several examples to illustrate how the results presented in the previous section can be helpful in the economic design of control systems.

Example 1 - Effect of Transmission Line Lag on a Reactor Under Direct Digital Control

Consider the following hypothetical problem in the design of a reactor. It is desired to control the temperature of a 200 gallon
<table>
<thead>
<tr>
<th>Equivalent Dead Time</th>
<th>Percent Increase in Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \left( \frac{T_0}{\tau_1} \right) + \frac{1}{2} \left( \frac{T_s}{\tau_1} \right) \left( \frac{T_s}{T_1} \right) ) (Continuous)</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \frac{T_0}{\tau_1} )</th>
<th>( \frac{T_s}{\tau_1} )</th>
<th>( \frac{T_s}{T_1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.20</td>
<td>330</td>
<td>330</td>
</tr>
<tr>
<td>0.30</td>
<td>198</td>
<td>198</td>
</tr>
<tr>
<td>0.40</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>0.50</td>
<td>108</td>
<td>108</td>
</tr>
<tr>
<td>0.60</td>
<td>87</td>
<td>87</td>
</tr>
<tr>
<td>0.70</td>
<td>71</td>
<td>71</td>
</tr>
<tr>
<td>0.80</td>
<td>62</td>
<td>60</td>
</tr>
</tbody>
</table>

Table 2.1. Comparing Actual Sampled Data Results with Results of Continuous Dead Time Approximation.
reactor by manipulating the jacket cooling water make-up rate (as shown in Figure 2.7). The control is to be implemented using digital computer located in a central control room about 700 feet from the reactor. At the present stage of design some question exists concerning how the valve position signal should be transmitted from the computer to the control valve, located at the reactor. Currently both pneumatic and electronic transmission are being considered, but it is not known which would be the best for this application.

Based on hardware cost alone it was determined that it would be cheaper to make the electronic to pneumatic conversion at the computer and run a pneumatic signal from the control room to the valve. However, it is feared that the lag associated with the long transmission line would reduce the effectiveness of the control system to such a degree that the more expensive electronic transmission line would be necessary.

Before the correct decision can be made the resulting degradation in system performance using a pneumatic line must be determined. Figure 2.4 provides a convenient way to readily determine this degradation. The only preliminary work which must be done is to evaluate the time constants in Figure 2.7-b.

Assuming the reactor is well mixed and the temperature drop across the jacket is low so that an average jacket temperature can be used, the relationship between reactor temperature and jacket temperature is a first order lag. The time constant can be expressed as:

\[ \tau_1 = \frac{M c_p}{U A} \]

where \( M \) = mass of reactor = 900#
Figure 2.7-a. Example 1 - Physical Description.

Figure 2.7-b. Example 1 - Mathematical Description.
\[ c_p = \text{heat capacity of reactor} = 0.7 \text{Btu}/\#^\circ \text{F} \]

\[ U = \text{heat transfer coefficient} = 95 \text{ Btu/hr ft}^2 \circ \text{F} \]

\[ A = \text{heat transfer area} = 200 \text{ ft}^3 \]

Therefore:

\[ \tau_1 = 0.0331 \text{ hr} = 1.99 \text{ min.} \]

The dead time of the process arises from the transport of cooling water from the point where the make-up water enters the system to some representative point inside the reactor. From the circulation rate of the jacket water, the length of the dead time is estimated to be approximately 20 seconds. Therefore, the dead time ratio is

\[ \Theta_0/\tau_1 = 20 \text{ sec}/(1.99)(60) \text{ sec.} = 0.167 \text{ min.} \]

For the electronic transmission line the transmission is essentially instantaneous and the valve action is very fast, therefore \( \tau_2 = 0 \). For the pneumatic transmission line the time constant is approximately 10 sec. Therefore, the lag ratio is

\[ \tau_2/\tau_1 = (10 \text{ sec})/(12.0 \text{ sec}) = 0.083 \]

Before Figure 2.4 can be used a performance criterion must be chosen. For this particular system, since large deviations in temperature will result in perhaps even larger deviations in the composition of the products, lets assume that ISE best describes the desired performance.

Assuming the sampling time to be 15 seconds (therefore \( T_s/\tau_1 = 0.125 \)) the correct dead time ratio to use in Figure 2.4-a is:
\[
\left( \frac{\theta_0}{\tau_1} \right)' = \left( \frac{\theta_0}{\tau_1} \right) + \left( \frac{1}{2} \right) \left( \frac{\tau_2}{\tau_1} \right) = .228
\]

The percent increase in error corresponding to \( \left( \frac{\theta_0}{\tau_1} \right)' = .228 \) and \( \left( \frac{\tau_2}{\tau_1} \right) = .083 \) is 88%. This means that for a step disturbance the pneumatic transmission line would yield 88% more error than would a corresponding electronic line. On the surface it would appear that the degradation would be fairly severe; however, the final decision on which is best will depend on the economic penalty associated with the increased error.

**Example 2 - Effect of Valve Speed on Composition Control Via Chromatograph.**

Two identical distillation columns are currently being controlled by manipulating the reflux ratio by a conventional temperature control scheme. Plans are underway to install a control system as shown in Figure 2.8. A single chromatograph will be shared between two columns. The question which arises concerns the control valve which is used to regulate the reflux rate. The valve which is currently in operation is rather slow (time constant - 4 sec.); however, since a shut down and modifications are already scheduled it would be convenient to install a faster valve (time constant = .5 sec) if significant improvement in system response will be recognized. Again, Figure 2.4 provides a convenient way to determine the actual improvement which could be realized.

The chromatograph which will be installed will require 20 seconds per analysis. It will alternate between the two columns at a rate of one cycle per minute (one analysis will be performed for both columns per minute). The results will be fed to a zero order hold which furnishes the input into a conventional proportional plus integral
Figure 2.8-a Example 2 - Physical Description.

Figure 2.8-b Example 2 - Mathematical Description.
controller (see Figure 2.8-b).

By frequency response techniques the transfer function relating overhead composition to reflux rate yields the following time constant and dead time.

\[ \tau_1 = 10 \text{ minutes} \]

\[ \theta_0 = 0.0 \]

The fact the dead time of the column is negligible might give the impression that Figure 2.4 does not apply; however, considerable dead time actually arises from the use of the chromatograph. Note that 20 seconds are required to process each sample and must be included as dead time in addition to one half the sampling time) to take into account the discrete nature of the control system). Therefore,

\[ \left( \frac{\theta_o}{\tau_1} \right)' = \frac{(0 + 25 \text{ sec.} + 30 \text{ sec.})/600 \text{ sec.}}{0.0915} \]

The hardware lag ratio for both the slow and the fast valve are

\[ \left( \frac{\tau_2}{\tau_1} \right) \text{ slow valve} = \frac{4 \text{ sec}}{600 \text{ sec}} = 0.00666 \]

\[ \left( \frac{\tau_2}{\tau_1} \right) \text{ fast valve} = \frac{0.5 \text{ sec}}{600 \text{ sec}} = 0.00834 \]

If we now look up the corresponding percent increase in error for each valve, we find that regardless of the performance criterion, the results are the same. The percent increase in error (over an hypothetical instantaneous valve) is very low in both cases. Therefore, with reasonable satisfaction we can say that the very slight (if any) improvement realized by the faster valve does not warrant the additional expense involved.
**Summary**

This chapter tries to help answer the general question concerning the effect of hardware lags in the control loop. The results of an extensive study of a generalized control loop are presented and can be easily applied in the selection and/or design of hardware elements such as control valves, pneumatic transmission lines, thermocouples and other actuating as sensing elements. The results are presented in such a way that hopefully the economic effect of a particular hardware element can be fairly accurately gauged and considered with other economic factors, which are more clearly determined.


Introduction

With the growing use of the digital computer in the process industries the control engineer is forced to deal with two fundamentally different control systems: the conventional analog loop and the more recent digital (sampled-data) control loop. From the standpoint of classical control theory the main theoretical difference between the two loops arises from the discrete nature of the digital controller. The analog controller operates continuously in determining the manipulated output whereas the digital makes adjustments only periodically.

The objectives of this chapter is an attempt to show how principles and practices for one of these types of control systems can be applied to the other. Specifically, it will be shown that many of the techniques (for example, controller tuning techniques and hardware lag effects) presently used for continuous control systems also can be applied directly to digital systems with only minor modification. This will be undertaken via an "equipment dead time" (1) for the sample and hold element. Thus, it is reasonable to begin by showing when a sample and hold element may be approximated by a dead time.

Development of Approximation

To focus on the problem and a possible solution consider the block diagram, shown in Figure 3.1, of a typical control loop found in the
Figure 3.1-a. Block Diagram of a Typical Digital Control Loop with a Sample and Zero Order Hold.

\[ G(s) = \frac{-\Theta s e^{-\Theta s}}{(1+\tau_1 s)(1+\tau_2 s)} \]

Figure 3.1-b. Block Diagram of an Equivalent Continuous Control Loop in Which \( G_a(s) \) Approximates the Behavior of the Digital Interface.
process industry. The loop consists of a feedback control algorithm (e.g., proportional plus integral) and a process described by a second order lag plus dead time. The problem is to find a continuous transfer function, \( G(s) \), which approximates the dynamic behavior of the sample and hold device located between the controller and plant in the discrete system (Figure 3.1-a). This would effectively reduce the sampled-data loop to an equivalent continuous loop (Figure 3.1-b).

Intuitively an inspection of the response of the sample and hold device provides a clue to one possible approximation. Figure 3.2 illustrates the response of sample and zero order hold to an arbitrary input. In most cases the hold is followed by a process with a large time constant which effectively smooths or averages the output. Note that the "average" output of the hold (shown by the dashed line in Figure 3.2) is simply the original signal delayed by one half the sampling time. When the "average" output of the hold is representative of the actual output it appears that a pure time delay (transportation lag) of one half the sampling time would adequately describe the behavior of the interface (i.e., \( G(s) = e^{-T/2)s} \).

Considering the behavior of the sample and hold device in the frequency domain provides some insight into the strengths and weaknesses of the dead time approximation. The open loop transfer function for a sample and zero order hold followed by an arbitrary process can be described as follows:

\[
HG^*(j\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} H(j\omega + jn\omega_s) G_p(j\omega + jn\omega_s)
\]  (3.1)
Figure 3.2. "True" and "Average" Response of a Zero Order Hold to an Arbitrary Input.
where

\[ H(j\omega) = \text{zero-order hold} \]
\[ G_p(j\omega) = \text{process} \]
\[ T = \text{sampling time} \]
\[ \omega_s = \text{sampling frequency} \]
\[ \omega = \text{arbitrary frequency} \]

The \( H(j\omega + j\omega_s) \) term can be evaluated by considering the Laplace transfer function for the zero order hold circuit:

\[ H(s) = \frac{1-e^{-sT}}{s} \] (3.2)

Substituting \( s = j\omega \) the transfer function can be written

\[ H(j\omega) = \frac{2e^{-\frac{j\omega T}{2}}}{e^{-\frac{j\omega T}{2}} - e^{-j\omega T}} \]

\[ = \frac{T \sin \left(\frac{\omega T}{2}\right) e^{-\frac{j\omega T}{2}}}{\frac{\omega T}{2}} \] (3.3)

Therefore:

\[ H(j\omega + j\omega_s) = \frac{T \sin \left(\frac{\omega + \omega_s}{2}\right) T}{\left[\frac{\omega + \omega_s}{2}\right] T} e^{-\frac{j(\omega + \omega_s)T}{2}} \] (3.4)

since \( T = \frac{2\pi}{\omega_s} \)
\[ H_{p}^{*}(j\omega) = \frac{1}{T} \sum \left[ \frac{T \sin \left( \frac{\omega}{\omega_s} + n\pi \right)}{\frac{\omega}{\omega_s} + n\pi} \right] e^{-\frac{\pi}{2}j(\omega + n\omega_s)T} G_p(j\omega + jn\omega_s) \]  

(3.5)

Note by assuming the angle \( \frac{\omega}{\omega_s} \) is small and assuming the process is insensitive to high frequency terms such as \( \omega_s \), the following simplification can be made:

1. When \( \frac{\omega}{\omega_s} \) is small, i.e., for frequencies much lower than the sampling frequency:

\[ \sin \left( \frac{\omega}{\omega_s} + n\pi \right) \approx 1 \]  

(3.6)

2. When \( G_p(s) \) is insensitive to the high frequency term \( \omega_s \):

\[ e^{-\frac{\pi}{2}(j\omega T + jn\omega_s T)} G_p(j\omega + jn\omega_s) \approx e^{-\frac{\pi}{2}j\omega T} G_p(j\omega) \]  

(3.7)

Therefore under these conditions the discrete transfer function reduces to the same pure dead time function previously developed:

\[ H_{p}^{*}(j\omega) \approx e^{-\frac{\pi}{2}j\omega T} G_p(j\omega) \]  

(3.8)

\[ H_{p}^{*}(s) \approx e^{-\frac{\pi}{2}Ts} G_p(s) \]  

(3.9)
On the surface the assumptions necessary in this development do not appear to be severe for the typical process operation. In general most plants found in the process industries are low pass filters by nature, and therefore they are not sensitive to the high frequency terms generated by the sampler (ωₕ). Also, under most operating conditions the fundamental frequency of the system response (ω) will be much lower than the sampling frequency (ωₛ); therefore, ω/ωₛ will usually be small.

To substantiate the validity of the above assumptions and evaluate the actual performance of the approximation, a typical plant was selected and studied both in the frequency and in the time domain. Figures 3.3-3.5 are the results of this study conducted with the second order plant,

\[ G_p(s) = \frac{1}{(\tau_p s + 1)^2} \]  

where \( \tau_p \) = time constant of the plant.

Illustrated are the open loop response, closed loop response (using a PI controller), and frequency response for the case \( \tau_p = 1.0 \).

In Figures 3.3 and 3.4 the open loop and closed loop response are illustrated at a rather large sampling time (T = \( \tau_p = 1.0 \)). Only this extreme value is shown because at lower, more reasonable sampling times the response of the actual discrete system and the response of the continuous approximation are essentially indistinguishable. Part of the reason the time response comparisons look so well can be understood by looking at the frequency response plots shown in Figure 3.5. The frequency response curves for this system essentially show that the primary
Figure 3.3. Comparison of Open Loop Response of a Discrete System with a Sample and Zero Order Hold to the Continuous System Using the Dead Time Approximation.

\[ G_p(s) = \frac{1.0}{(1+\tau_p s)^2} \]

\[ H(s) = \frac{1-e^{-sT}}{s} \]

\[ \tau_p = T = 1.0 \]
Figure 3.4. Comparison of Closed Loop Response of a Discrete System with a Sample and Zero Order Hold to the Continuous System Using a Dead Time Approximation.

\[
G_p(s) = \frac{1.0}{(1+\tau_s)^2} \quad \text{Process}
\]

\[
G_c(s) = K_c (1 + \frac{1}{T_1 s}) \quad \text{Controller}
\]

\[
H(s) = \frac{1-e^{-Ts}}{s} \quad \text{with} \quad T = \tau_p
\]

Response of Discrete Loop

Response of Corresponding Continuous Loop Using Dead Time Approximation
Figure 3.5. Comparison of Frequency Response of a Discrete System with a Sample and Zero Order Hold to the Continuous System Using a Dead Time Approximation.
factor determining the effectiveness of the approximation is the "fold-over" frequency (predicted by Shannon's sampling theorem) which says that the sampling rate must be at least twice the highest frequency component of the response. If sampling is below this critical frequency the frequency response "folds over" and becomes distorted as is vividly demonstrated in Figure 3.5. Below this critical frequency the difference between the discrete system and the continuous approximation is very low in both the magnitude plot and in the phase plot. Above the critical frequency the difference is quite large as would be expected.

From this study the possibility of using the dead time approximation looks indeed encouraging at least for reasonably low sampling times and under normal operating conditions. The next logical question which should be answered is what generalizations can be made about the discrete system using this simplification. The remainder of this chapter will concern several possible generalizations. Each will be discussed both to more clearly define the general effectiveness of the approximation and to illustrate areas where the approximation could be valuable.

Applications and Examples

Assuming the above approximation is acceptable, a study of the control loop shown in Figure 3.1-a is greatly simplified. First, the fact that the discret problem is reduced to an equivalent continuous problem is encouraging, but perhaps more important than this simplification is the fact that in loops already containing a dead time (quite common), the number of parameters required to describe the system is reduced by one. Note in Figure 3.1-a, if the controller used is a two
mode controller (i.e., \( G_c(s) = K_c \left(1 + \frac{1}{T_1s} \right) \)) the number of parameters required to specify the discrete system is as follows:

\[
\begin{align*}
T_1, K_c & \quad \text{Controller parameters} \\
\tau_1, \tau_2 & \quad \text{Process time constants} \\
\theta & \quad \text{Process dead time} \\
T & \quad \text{Sampling time}
\end{align*}
\]

In all a total of six parameters are required for the discrete case, but, for the equivalent continuous case, by combining the dead time of the plant and the resulting dead time of the approximation into an equivalent dead time \((\theta' = \theta + \frac{T_s}{2})\) only five terms are needed to completely describe the system.

Perhaps one of the most fruitful areas of application of the approximation lies in controller tuning. Much work has been done in the past in the area of tuning the one, two, and three mode continuous controllers. Numerous procedures have been developed based on the first order lag plus dead time model of the process \((2), (3), (4), (5), (6), (7)\). One of the better tuning relations which is developed to yield minimum IAE (Integral of the Absolute Error) is given below for both the proportional \((G_c(s) = K_c)\) and proportional plus integral \((G_c(s) = K_c \left(1 + \frac{1}{T_1s} \right))\) algorithm \((7)\):

Proportional:

\[
K_c K = 0.9023 \frac{\theta}{\tau_p}^{-.985}
\]

\[(3.11)\]
Proportional + Integral:

\[ K_{c} = 0.984 \begin{pmatrix} 0.0 \frac{\theta}{\tau} \end{pmatrix} \]

\[ \tau_{p}/T_{i} = 0.608 \begin{pmatrix} 0.0 \frac{\theta}{\tau} \end{pmatrix} \]

where \( K_{c} \) = Controller Gain

\( T_{i} \) = Controller Reset

\( K \) = Process Gain

\( \tau_{p} \) = Process Time Constant

\( \theta \) = Process Dead Time

Little effort has been made to extend the use of these and other well developed relationships to the sampled data case other than for very fast sampling rates, at which the continuous tuning parameters can be directly applied to the discrete control system (9). Based on the demonstrated effectiveness of the dead time approximation under normal operating conditions it would appear to be a very good way to extrapolate continuous tuning to the sampled data case. Note, all that would be required to use the above equations for a discrete controller is to define an equivalent dead time \( \theta' \) where,

\[ \theta' = (\theta + T/2) \]

\( \theta \) = actual dead time of process

\( T \) = sampling time

To test the validity of such an extrapolation an extensive study of
sampled data controller tuning recently conducted (8) was considered and the approximation was used and the results compared to the findings of this study. Figure 3.6 (proportional controller) and Figure 3.7 (proportional plus integral controller) illustrate the tuning parameters found by this study (using minimum IAE) at several values of sampling times including the case of continuous sampling. The solid lines illustrate the corresponding parameters predicted by using only the results of the continuous plot. Note in both figures at reasonably low values of sampling time (T ≤ .4) the difference between true optimum controller settings and settings predicted by the dead time approximation is small, and in fact, they are essentially zero at relatively large values of dead time. In general, even in the region where the agreement is the poorest (low dead times and low sampling rates), the percent error is still low and probably within the actual accuracy of the model parameters.

Another successful application of the continuous approximation is in predicting the effect of hardware lag on system performance in a sampled data environment. This use is demonstrated by the results of an extensive study conducted in both the continuous and discrete case (see Chapter II).

The study considered a system consisting of a PI controller, \( G_c(s) = K_c (1 + 1/T_1 s) \), a hardware element, \( G_v(s) = 1/(1 + \tau_v s) \); and a process, \( G_p(s) = e^{-\theta_s} / (1 + \tau_p s) \). The control system was subjected to a disturbance for various combinations of \( \tau_v / \tau_p \), \( \theta / \tau_p \) and \( T / \tau_p \) (where \( T \) = sampling time). For each combination the system performance was evaluated in terms of the integral of the absolute error, defined:
Figure 3.6. Comparison of Optimum Proportional Controller Gains of a Discrete System with Gains Predicted by Dead Time Approximation.
Figure 3.7-a. Comparison of Optimum PI Controller Gains of a Discrete System with Gains Predicted by Dead Time Approximation.
Optimum Parameters for Discrete (8) System

Parameters Predicted Using Dead Time Approximation

Figure 3.7-b. Comparison of Optimum PI Controller Reset of a Discrete System with Reset Predicted by Dead Time Approximation.
\[ IAE = \int_0^\infty |x-SP| \, dt \tag{3.14} \]

where \( x \) = system output
\( SP \) = set point

The results are presented in terms of a % increase in error defined as:

\[ \text{% Increase in Error} = \frac{IAE \times 100}{IAE @ (\frac{\tau_v}{\tau_p} = 0)} \left| \begin{array}{c}
\text{const} \\
\frac{T}{\tau_p}, \frac{\theta}{\tau_p}
\end{array} \right| \tag{3.15} \]

Figure 3.8 shows two such "% Increase in Error Plots", taken at \( T/\tau_p = 0.2 \) and \( T/\tau_p = 0.0 \). The two plots are shifted by a dead time at 0.2 and superimposed to illustrate the effect of the approximation.

Note the plots are almost identical at all combinations of \( \frac{\tau_v}{\tau_p} \) and \( \frac{\theta}{\tau_p} \). Comparison of plots at sampling times greater than 0.2 does not give quite as good results; however, even at sampling times as large as 0.4 the only noticeable deviation occurs at low values of \( \frac{\theta}{\tau_p} \) where the plots are the most sensitive.

The net result is that with reasonable accuracy the entire family of % Increase in Error plots (one for each possible sampling time) can be described by a single plot. To predict the effect of the hardware lag at any sampling frequency it is necessary only to define, as before, an effective dead time \( (\theta' = \theta + T/2) \) and use the continuous plot directly.

Finally, one area in which the approximation appears to be very poor is in the area of closed loop stability. Figure 3.9 presents an
Optimum Parameters for Discrete (8) System

Parameters Predicted Using Dead Time Approximation

Figure 3.7-b. Comparison of Optimum PI Controller Reset of a Discrete System with Reset Predicted by Dead Time Approximation.
\[ IAE = \int_{0}^{\infty} |x - SP| \, dt \quad (3.14) \]

where \( x \) = system output

\( SP \) = set point

The results are presented in terms of a \( \% \) increase in error defined as:

\[ \% \text{Increase in Error} = \frac{IAE \times 100}{IAE @ (\tau_v/\tau_p = 0)} \quad (3.15) \]

\[ \text{const} \]

\[ T/\tau_p, \Theta/\tau_p \]

Figure 3.8 shows two such "\( \% \) Increase in Error Plots", taken at \( T/\tau_p = 0.2 \) and \( T/\tau_p = 0.0 \). The two plots are shifted by a dead time at \( 0.2 \) and superimposed to illustrate the effect of the approximation. Note the plots are almost identical at all combinations of \( \tau_v/\tau_p \) and \( \Theta/\tau_p \). Comparison of plots at sampling times greater than 0.2 does not give quite as good results; however, even at sampling times as large as 0.4 the only noticeable deviation occurs at low values of \( \Theta/\tau_p \) where the plots are the most sensitive.

The net result is that with reasonable accuracy the entire family of \( \% \) Increase in Error plots (one for each possible sampling time) can be described by a single plot. To predict the effect of the hardware lag at any sampling frequency it is necessary only to define, as before, an effective dead time (\( \Theta' = \Theta + T/2 \)) and use the continuous plot directly.

Finally, one area in which the approximation appears to be very poor is in the area of closed loop stability. Figure 3.9 presents an
Figure 3.8. Comparison of True Percent Increase In Error Due to Increased Hardware Lag at Sampling Time = 0.2 to the Percent Increase in Error Predicted by the Dead Time Approximation.
Figure 3.9. Comparison of Ultimate Gain of a Discrete System with Ultimate Gain Predicted by Dead Time Approximation.
ultimate gain comparison for the discrete and the corresponding continuous proportional control of a pure first order system. Note the ultimate gain of the discrete system is always lower than the gain predicted by the continuous approximation. The fact the approximation is not particularly good in this application is not surprising considering the unusually high frequency the system response is forced to undergo. One assumption fundamental to the development of the approximation is that the sampling frequency be much larger than the fundamental frequency of the system response. This assumption is obviously poor under these extreme conditions.

Summary

In summary, except at very high sampling times (relative to the time constant of the system) or when under extreme operating conditions (such as operating at or near the stability limits of the system), a discrete control system containing a sampler and zero order hold can be successfully approximated by an equivalent continuous system containing a pure dead time of one half the sampling time. The value of the approximation lies in the fact it provides a basis for applying many of the techniques currently used only in continuous control systems to the discrete control system as well. Two successful techniques are demonstrated in this chapter (controller tuning and predicting the effect of hardware lags) along with an unsuccessful technique (stability limits of a control system). Hopefully the success and limitations of the approximation demonstrated in this article will provide the basis for further generalization of the techniques and methods used in both discrete control and in continuous control.
LITERATURE CITED


CHAPTER IV

CONTROL OF A HIGH ORDER PLANT USING A TIME-OPTIMAL SECOND ORDER SWITCHING CURVE

Introduction

The application of minimum time control to a large class of chemical processes provides several unique problems not normally encountered in other fields (2), (3), (4). The nature of the problems essentially rules out any straight-forward application of the minimum principle (5) and should be at least recognized before any workable control strategy can be designed utilizing the potential of optimal control. This chapter will discuss some of the practical problems involved with directly applying minimum-time control and will offer a workable design of a "time-optimal" controller, applicable to the class of chemical processes under consideration.

The processes which will be considered in this chapter will be the complex large scale chemical processes commonly found in the chemical industry. Typically these processes can be described as high-order, over-damped systems usually operating within limits on the allowable control action. Due to the dynamic nature of the environment this control system is usually required to follow periodic changes in the set-point as well as to correct for numerous disturbances which frequently enter the loop. To meet these dual needs some form of feedback compensation has typically been used.
With the use of the versatile but expensive digital control computer in the chemical process industries there is an increased demand to replace conventional control strategies with more sophisticated techniques in cases where some real economic advantage can be gained. The class of chemical processes to be considered in this article offers one such area. The one-, two-, or three-mode analog controllers conventionally used in the control of most chemical processes perform well in most applications; however, for the high-order over-damped processes under consideration their effectiveness rapidly deteriorates when the manipulated variable becomes saturated. The narrower the operating limits of allowable control action, the less effective will be the conventional controller. Replacing the conventional controller with a time-optimal controller, taking into consideration all the constraints, appears to be a systematic approach to handling the problem. However, several problems arise due to the nature of the environment in which the control system must operate.

Complex Solution

The high order of the system poses one major problem. Like any other high order system, the exact solution of the necessary equations prescribed by the minimum principle can be rather complex. In general for an \( n \)th order linear system the exact solution requires at best an iterative solution of an \( n \)th order two-point boundary value problem. Such a solution is in general possible; however, for several reasons it is not particularly applicable to the closed loop environment in which it must operate.

The most obvious reason is simply the shear computation power and time required for each solution. If the system is operating in a true
closed loop fashion (which seems almost necessary to satisfy the regulator function of the desired control system) the problem must be resolved at every sampling instant. Granted, it is possible that in some situations the computer solution may be fast enough that for very slow processes the control could be implemented in this fashion, but it is at a sacrifice of both computer storage and time which would normally be available to other process loops or other control functions.

Methods have been developed to determine a non-linear controller based on switching curves so that time optimal control can be applied in a feedback manner without having to re-solve the problem at every sampling instant. The difficulty involved with implementing this idea to the problems under consideration is for high order systems the physical significance of the switching curve is lost and switching is based on the solution of n-1 highly nonlinear equations with n unknowns and several inequality constraints. Once again, for high order systems much of the capacity of the process computer would be required for this single control function.

Unmeasurable State Variable

Another serious problem encountered in using the exact solution in a feedback manner is the need for information on all the states of the system. In most processes the states of the system do not correspond to any physical variable which could be directly measured, and consequently, they must be determined from the output of the process. Even using sophisticated filtering techniques the results are poor for high-order states. A knowledge of the complete state of the system is essential if an exact solution is to be implemented in a feedback fashion. The lack of such knowledge yields an exact solution impossible except in
the open loop fashion which has already been deemed unsatisfactory.

Model Errors and Simplifications

In considering the problems involved in obtaining an exact solution, the fact should be recognized that even the best system model is probably an over-simplification of reality. The chemical process is in general unlike other systems in which the basic mathematical description corresponds directly to the physics of the problem. Typically a model of the process is determined by empirical means such as frequency response analysis or by a nonlinear regression on a set of time response data. The result usually obtained is a linear transfer function which is assumed to be the model of the system. The actual process may be either a very high-order system or a distributed parameter system for which the model is only a "good" approximation. The real process may also contain any number of nonlinearities not included in the model. The net result of all the errors and simplifications in the process model may or may not be significant; however, they do render hopeless the possibility of a truly exact solution. Regardless of the amount of time and effort spent to implement an exact solution it turns out to be based on a process model which does not describe the system exactly.

Unknown Equilibrium

Another difficulty which arises in using the "exact" solution is the fact that the complexity of the problem may be such that the equilibrium point of the manipulated variable may not be known and might be impossible or impractical to determine (such as the steam rate to the reboiler of a distillation column). Also, due to the dynamic nature of the process environment, the actual equilibrium point may actually be changing with time.
To be concerned about an exact solution when the proper stationary point is not known would seem to be a gross waste of time and effort.

In light of the problems discussed, the possibility of time optimal control appears remote. A feedback design is almost essential, not only to correct for unknown disturbances but also to compensate for modeling errors and errors in the estimated stationary position of the system. Due to the high order of the process under consideration any direct implementation of an exact solution in this manner would be, to say the least, costly in terms of computer time and memory. The logical route is to abandon the idea of an exact time optimal controller and make several simplifications which would yield a workable and hopefully a "near-optimal" controller. The remainder of this chapter will be concerned with two such simplifications based on a second-order approximation to the system equations.

**Second Order Feedback**

Considerable simplification is obtained if the system equations can be assumed to be second order. One computational advantage lies in the need to calculate only one additional state from the system output, but by far the greatest advantage lies in the simplicity of the feedback control scheme. Figure 4.1 illustrates such a control scheme. Note the controller consists of three elements: a filter to determine the derivative of the output, a nonlinear switching curve and a bang-bang element, all of which can be implemented easily on the digital computer.

The key to the entire system is the switching curve. The switching curve can be determined directly from the second-order model with no direct reference to the minimum principle as such. The system equation need only be solved in reverse time to determine all the initial
Figure 4.1 Time Optimal Controller for a Second Order Plant.
conditions which lead to the origin for both the maximum and the minimum control. For the second-order system,

\[ y(t) + (\tau_1 + \tau_2) \frac{dy(t)}{dt} + \tau_1 \tau_2 \frac{d^2 y(t)}{dt^2} = u \]  

(\tau_1 \text{ and } \tau_2 \text{ are real constants and } u \text{ is the control action which can assume to be either } u_{\text{max}} \text{ or } u_{\text{min}}.) \text{ The general solution is as follows for both the case where } \tau_1 \neq \tau_2 \text{ and the case } \tau_1 = \tau_2 = \tau:

Case I \quad \tau_1 \neq \tau_2

\[ y(t) = \frac{(\xi_2 \tau_2 + \xi_1 - U)}{(\tau_2 - \tau_1)} e^{-t/\tau_1} + \frac{(\xi_2 \tau_1 + \xi_1 - U)}{(\tau_2 - \tau_1)} e^{-t/\tau_2} + U \]  

\[ \frac{dy(t)}{dt} = \frac{(\xi_2 \tau_2 + \xi_1 - U)}{(\tau_2 - \tau_1)} e^{-t/\tau_1} - \frac{(\xi_2 \tau_1 + \xi_1 - U)}{(\tau_2 - \tau_1)} e^{-t/\tau_2} \]  

Case II \quad \tau_1 = \tau_2 = \tau

\[ y(t) = (\xi_1 - U)(1+t/\tau) + \xi_2 t \cdot e^{-t/\tau} + U \]  

\[ \frac{dy(t)}{dt} = (\xi_1 - U)(1+t/\tau)(1/\tau) + \xi_2 t/\tau \cdot e^{-t/\tau} \]  

where \( \xi_1 \) = initial conditions on \( y(t) \)

\[ \xi_2 \] = initial conditions on \( \frac{dy(t)}{dt} y(t) \)

In determining the switching curve for Case I it is convenient to solve the equations for \( \xi_2 \) and \( \xi_1 \) with \( y(t) = 0 \) and \( \frac{dy(t)}{dt} = 0 \). After some
algebraic manipulations time can be eliminated from the resulting equations to yield the following switching curve (5):

\[ x_2 = \frac{x_1}{|x_1|} \left( 1 + |x_1|^{\alpha} - 1 \right) \]  \hspace{1cm} (4.6)

where

\[ \alpha = \frac{\tau_1}{\tau_2} \quad (\tau_2 > \tau_1) \]  \hspace{1cm} (4.7)

\[ X_1 = (\xi_2 + U)e^{t/\tau_1} - U \]  \hspace{1cm} (4.8)

\[ X_2 = (\xi_2 + U)e^{t/\tau_2} - U \]  \hspace{1cm} (4.9)

\[ U = \pm 1 \]  \hspace{1cm} (4.10)

For Case II the nonlinearity of the equations prevents a functional form of the switching curve. However, the curve can be easily implemented by solving the equations in reverse time for \( \xi_1 = 0 \) and \( \xi_2 = 0 \) and storing the values in a table.

\[ y_1 = U \frac{t}{\tau} - 1 \left( e^{t/\tau} + 1 \right) \]  \hspace{1cm} (4.11)

\[ y_2 = y_1 = -U \frac{t}{\tau} e^{t/\tau} \]  \hspace{1cm} (4.12)

Second-Order Approximation

In an attempt to utilize the simplicity of the second-order time-optimal controller the obvious procedure would be to fit a second-order model to the higher order system and determine the time constant
necessary to construct a second-order switching curve. To test the effectiveness of such an approximation the following 6th order process was selected:

\[ G(s) = \frac{1}{(1 + s)^6} \quad (4.13) \]

In order to determine the second-order system which gives the "best fit" a nonlinear regression was used to fit the step response over the range of time \( t = 0 \) to \( t = 20 \), giving the results \( \tau_1 = \tau_2 = 3.032 \). From Figure 4.2 it is obvious the second-order approximation is poor, at least in open-loop performance. Figure 4.3 and Figure 4.4 illustrating the behavior of the control loop both in the time domain and in the phase-plane, only confirm the suspicion about the closed-loop performance of the control system. Note a rather severe limit cycle is reached with the magnitude of oscillation almost equal to the original displacement. It is evident that a simple second-order approximation is not suitable for the high-order process being considered.

**Second-Order Plus Dead Time Approximation**

From observation of the transient response data it appears that a much better approximation could be obtained if a pure dead time is included with the second-order model. Indeed, after performing a nonlinear regression, the time response fit appears to be very good (see Figure 4.5). The only problem is how to treat the dead-time. The second-order time optimal control discussed up to now has not included any nonlinear term in the system equations.

One convenient way to eliminate the problem is to use the Smith predictor (1), (6). The Smith predictor is designed to effectively
Process:

\[ G(s) = \frac{1}{(s+1)^6} \]

Model:

\[ G_m(s) = \frac{1}{(\tau_1 s+1)(\tau_2 s+1)} \]

\[ \tau_1 = 3.032 \]

\[ \tau_2 = 3.032 \]

Figure 4.2. Second-Order Model - Minimum Error Squared Fit.
Figure 4.3. Time Response: Time Optimal Control of a 6th Order Plant Using a Second Order Model.
Figure 4.4. Phase Plot: Time Optimal Control of a 6th Order Plant Using a 2nd Order Model.
Model:
\[ G_m^2(s) = \frac{-\theta_o s}{(\tau_1 s + 1)(\tau_2 s + 1)} \]

\[ \theta_o = 2.38 \]
\[ \tau_1 = 1.851 \]
\[ \tau_2 = 1.851 \]

Process:
\[ G(s) = \frac{1.0}{(1 + s)^6} \]

Figure 4.5. Second Order Plus Dead Time Model - Minimum Error - Squared Fit.
eliminate dead time from a typical process loop in which the process transfer function and the dead time are precisely known. This technique includes a process and dead time model in the controller in such a way that the net effect is to eliminate the dead time from the loop dynamics. Figure 4.6-a shows in block diagram form how the predictor is implemented. The closed loop solution is:

\[
Y(s) = \frac{G_c(s)G_p(s)e^{-sT_p}}{1 + G_c(s)G_p(s) + G_c(s)G_p(s)e^{-sT_p} - G_c(s)G_m(s)e^{-sT_m}} \quad R(s)
\]

(4.14)

where

- \( G_c(s) \) = control algorithm
- \( G_p(s) \) = process transfer function
- \( T_p \) = process dead time
- \( G_m(s) \) = model transfer function
- \( T_m \) = model dead time

Note in the ideal case \( G_p(s)e^{-sT_p} = G_m(s)e^{-sT_m} \) the solution simplifies to (see Figure 4.6-b):

\[
Y(s) = \frac{G_c(s)G_p(s)e^{-sT_p}}{1 + G_c(s)G_p(s)}
\]

(4.16)

The reduced expression does not contain a dead time term in the characteristic equation.
Figure 4.6-a. Compensating for Dead Time Using a Smith Predictor

Figure 4.6-b. Equivalent Block Diagram Using a Perfect Smith Predictor ($G_m(s) = G_p(s); T_m = T_s$).
In applying this technique to the time optimal feedback control the philosophy is to effectively reduce the order of the control system by subtracting out the dead time (Figure 4.7). Note that the actual process does not contain a dead time. However since the process behaves much like a second-order plus dead time system it is hopeful that sufficient "cancellation" will result that the overall system behaves much like a second-order system.

Figure 4.8 and 4.9 show the results of such a control system. In light of how poor the results were for the pure second-order approximation, the results appear to be very good. The system output was, in an encouragingly short period of time, brought and held close to the desired value. Figure 4.9 is the phase plot of the predictor output. Note the limit cycle has considerable magnitude; however, it should be noted that this is not the actual system output but the output of the predictor. The limit cycle is not noticeable in the actual output of the system.

Another study was made using the second-order lag plus dead time approximation over the region of the step response from $t=0$ to $t=5$ (instead of $t=20$). Figure 4.10 illustrates the results, and note the fit is close over the first part of the curve but poor for the remainder.

The system response in Figures 4.11 and 4.12 show the results obtained. Note the response has a much faster initial rise time but a longer settling time. In fact it looks very similar to the quarter delay ratio often used as a basis for tuning conventional controllers. This approximation also appears to be a successful control strategy.
Sw itching Curve
Based on $\tau_1$ and $\tau_2$

Figure 4.7. Smith Predictor with Time Optimal Controller.
Figure 4.8. Time Response: Time Optimal Control of a 6th-Order Plant Using a 2nd Order Plus Dead-Time Model.
Figure 4.9. Phase Plot of Predictor Output: Time Optimal Control of a 6th Order Plant Using a 2nd Order Plus Dead Time Model.
Figure 4.10. Second Order Plus Dead Time Model - Minimum Error Squared Fit.
Figure 4.11. Time Response: Time Optimal Control of a 6th Order Plant Using a 2nd Order Plus Dead Time Model.

\[ G(s) = \frac{1.0}{(1+s)^6} \]

\[ G_m(s) = \frac{e^{-1.8s}}{(2.241s+1)(2.663s+1)} \]
Figure 4.12. Phase Plot of Predictor Output: Time Optimal Control of a 6th Order Plant Using a 2nd Order Plus Dead Time Model.
Summary

This investigation has shown that the optimal control strategy derived from a simple model can be successfully used to control a more complex process. This is attractive as (1) a detailed model need not be developed, (2) the computations involved would not burden the computer, (3) equilibrium points need not be determined accurately, and (4) the hardware needed to implement this procedure is simple. Techniques such as this appear to be one route by which optimal control theory can be applied to process systems.


CHAPTER V

FORMULATING THE NONLINEAR LEAST-SQUARE MODEL
REGRESSION FOR FAST ON-LINE ANALYSIS

Introduction

It is becoming widely accepted that the installation of a digital computer for process control cannot be easily justified on the economics of simply replacing conventional analog equipment. More and more emphasis is being placed on exploiting the flexibility and logic capabilities of the digital computer by implementing advanced control concepts such as plant setpoint optimization, non-interacting control of multivariable plants, feedforward control, cascade control, nonlinear control, etc.

One common requirement for all the above techniques is a reasonably accurate mathematical description of the process under control. If the process model is poor, the full potential of the techniques mentioned above will not be realized. This problem is even more complicated by the fact that many processes encountered in the process industries are non-stationary, that is, the "correct" or "best" model actually changes with time due to changes in such factors as operating level, catalyst age, environmental conditions, etc. As a result some form of adaptive control is needed to measure the changes in process parameters and compensate by adjusting the control parameters.

Numerous techniques have been developed to adapt to changes in process parameters. The most general of the techniques is via a direct model determination which consists of three steps:
(1) periodic data collection
(2) determination of new model parameters from process data
(3) updating of control parameters based on the new model parameters

This technique is very attractive in a complex system which contains numerous loops and several levels of control, all of which might be controlled by a different algorithm or strategy. In such cases a single regression program could be used in conjunction with all the loops and control levels to adapt to changes in process parameters.

This chapter will be primarily concerned with the first two steps in this approach to adaptive control. The problems associated with step 3, updating control settings for a given set of process parameters, are fairly straightforward in most cases. Most advanced control concepts are designed around some assumed process model; and once the parameters of the model are known, the controller is specified (1), (2). Even for the classical one-, two-, and three-mode controllers, numerous tuning tables and equations are available which present the gain, reset, and rate parameters as functions of sampling time and model parameters (5), (6), (7), (8), (9), (10).

The data collection and model regression steps are not so well-developed. Long data tables are often required to collect and store the data and the program which performs the regression is generally long and time-consuming. This article will discuss this problem and present a general way to formulate the data collection and parameter regression to greatly reduce storage requirements and running time over conventional "off-line" programs and free the computer to perform additional tasks which perhaps could not be easily implemented using conventional
analog equipment.

"Least-Square" Regression

One popular technique for both on-line and off-line parameter identification is simply a "least-square" fit of either the actual process output (8), the derivative of the process output (4), or the integral of the process output (4). This technique is very general and can be employed with no great difficulty with most models commonly used to describe chemical processes. It is statistical by nature and is particularly suitable for process data which is typically accompanied by noise. It is easily understood and can be easily programmed; however, it should be noted that there is a considerable difference in program requirements for an effective on-line program as opposed to the more conventional off-line program. For a program which is operating in a real time environment in which many different operations must be essentially performed simultaneously, computation time is at a premium. Furthermore, the numerous functions a single computer is expected to perform, coupled with a rather limited core size, demand that the length of all on-line programs and data tables be minimized. While it would be nice to have a fast, efficient off-line program, it is not so essential. The premium for an off-line program is probably more properly placed on flexibility and generality. These distinctions are fairly obvious; however, when the time comes to initiate on-line identification the general tendency at present is to essentially use the off-line version of the program. If care is taken in formulating the identification program specifically for on-line use, significant improvements can be made both in terms of execution time and core storage.
Development of a Nonlinear "Least-Square" Regression

To illustrate the difference between a conventional regression program and one written specifically for on-line use, consider the development of a nonlinear least-square program to fit a set of input-output data to a second-order lag plus dead time model of the form:

\[ G_m(s) = \frac{-\theta s}{(1 + \tau_1 s)(1 + \tau_2 s)} \]  

(5.1)

Process Parameters

- $K$ = gain
- $\tau_1, \tau_2$ = time constants (real, distinct)
- $\theta$ = dead time (real)

The first step in developing a regression program is to define a criterion function or measure of fit by which to judge which parameter values best fit the data. The criterion function used by all "least-square" programs is defined as follows:

\[ E = \sum_{i=1}^{M} (x_i - \hat{x}_i)^2 \]  

(5.2)

where
- $E$ = fit error
- $x_i$ = actual process output at point $i$
- $\hat{x}_i$ = predicted process output at point $i$
- $M$ = total number of data points

This expression can be expanded by expressing $\hat{x}_i$ in terms of the pulse
transfer function corresponding to the dynamic model.

\[ \hat{x}_i = C_1 x_{i-1} - C_2 x_{i-2} + B_1 u_{i-(n+1)} - B_2 u_{i-(n+2)} + B_3 u_{i-(n+3)} \]  
(5.3)

or

\[ E = \sum_{i=1}^{M} x_i - C_1 x_{i-1} + C_2 x_{i-2} - B_1 u_{i-(n+1)} + B_2 u_{i-(n+2)} + B_3 u_{i-(n+3)} \]  
(5.4)

The constants in the above equation are functions of the parameters of the system \((\tau_1, \tau_2, \Theta, \text{ and } K)\) defined by the following equations:

\[ C_1 = e^{-T/\tau_1} + e^{-T/\tau_2} \]  
(5.5)

\[ C_2 = \begin{pmatrix} e^{-T/\tau_1} & e^{-T/\tau_2} \\ e^{-T/\tau_2} & e^{-T/\tau_1} \end{pmatrix} \]  
(5.6)

\[ B_1 = K \left[ \frac{1}{\tau_1} - \frac{1}{\tau_2} \right] + \begin{pmatrix} -mT/\tau_1 & -mT/\tau_2 \\ e^{-T/\tau_2} & -e^{-T/\tau_1} \end{pmatrix} \]  
(5.7)

\[ B_2 = K \left[ \frac{1}{\tau_1} - \frac{1}{\tau_2} \right] \begin{pmatrix} e^{-T/\tau_1} & e^{-T/\tau_2} \\ 1+e^{-T/\tau_1} & 1+e^{-T/\tau_2} \end{pmatrix} + \begin{pmatrix} -T/\tau_1 & -T/\tau_2 \\ e^{-T/\tau_2} & -e^{-T/\tau_1} \end{pmatrix} \]  
(5.8)

\[ B_3 = K \left[ \frac{1}{\tau_1} - \frac{1}{\tau_2} \right] \begin{pmatrix} e^{-T/\tau_1} & e^{-T/\tau_2} \\ e^{-\tau_2/T_1} & e^{-\tau_1/T_2} \end{pmatrix} + \begin{pmatrix} -T/\tau_1 & -T/\tau_2 \\ e^{-T/\tau_2} & -e^{-T/\tau_1} \end{pmatrix} \]  
(5.9)
where

\[ T = \text{sampling time} \]
\[ n = \text{integer delay (integer of } \theta/T) \]
\[ m = 1 - \theta/T - n \]

\[ \text{note } \theta = (1-m)T + n \]

The object of this regression is to find values of \( \tau_1, \tau_2, \theta, \) and \( K \) which give the minimum fit-error, \( E^* \), subject to the constraints that \( \tau_1, \tau_2 \) and \( \theta \) are positive and real.

\[
E^* = \min_{\tau_1, \tau_2, K, \theta} \sum_{i=1} \left( x_i - C_1 x_{i-1} + C_2 x_{i-2} - B_1 u_{i-(n+1)} + B_2 u_{i-(n+2)} + B_3 u_{i-(n+3)} \right)
\]

\[(5.10)\]

In developing an off-line regression program, the formulation is essentially complete. The minimization indicated above can be easily implemented by performing a direct search of \( \tau_1, \tau_2, K, \) and \( \theta \) to find \( E^* \) by the use of one of the many multidimensional search techniques currently available (12). In certain other cases, this can be reduced to a one-dimensional search (see Chapter VI). No restrictions have been imposed on the parameter regression, and virtually any type of data, given enough data points, could be analyzed (including closed loop data).

However, consider the program requirements both in terms of computation time and core storage. One characteristic of even the better search techniques is the requirement of a large number of functional evaluations (number of times the "fit-error" must be determined) before the optimum parameters can be located. This becomes significant in terms of
computation time when the equations defining the fit-error are considered. For each functional evaluation the computation indicated in Equations (5.4-5.9) must be performed. Note the summation indicated in Equation (5.4) contains \( M \times 6 \) terms and could be quite time consuming if \( M \), the number of data points, is reasonably large (as is generally demanded to insure a good estimate of the parameters). In terms of core storage the requirements for program logic alone are not severe, provided a reasonably simple search technique is selected. However, in order to carry out the regression the entire input-output data must be stored. Therefore, two long data tables are required and contribute considerably to the total program requirement. The demands made on computer time and storage are often very large with the regression program formulated as shown above. However, for off-line analysis the demands are not serious, and in fact they are a relatively cheap price to pay for generality. For on-line regression on the other hand the formulation is very costly.

Data Collection

Before investigating the possibilities of modifying the off-line program to emphasize speed and efficiency, consider (from a programming point-of-view) how data might typically be obtained for on-line regression. Figure 5.1 illustrates a typical approach to the problem. A master program initiates and directs the data collection. Upon either an operator request or a periodic command the program would select the appropriate loop and check operating conditions. If operating conditions are within limits the appropriate controller would be placed on manual and a pulse of height \( H_p \) and duration \( T_p \) would be generated in
Master program to initiate and control data collection. It selects the appropriate loop, sets the controller on manual, generates a pulse in the controller output, holds the controller on manual for a specified period of time, and notifies the regression program when the data collection is complete.

Figure 5.1. Micro-Flow Chart for On-Line Data Collection.
the manipulated variable for the respective loop. A separate program would store the resulting data collected during the test period for later use by the regression program. It should be noted that there is a subtle difference between data collected in this manner and data collected for off-line identification. In off-line identification the purpose is to identify the parameters of a particular process (such as a reactor in which a model is desired to describe the reactor temperature as a function of the flow rate of the jacket make-up water). Typically, special sensors and recorders are needed to convert the process variables to data which can be used by the off-line program. Data is collected on-line from a different point-of-view. It includes not only the response of the process to changes made in some process variable (i.e. make-up water rate), but also includes the dynamics of the actuator, the sensor, transmission lines and the digital interface (A/D and D/A). Not only does this model provide a much more accurate description of what the actual controller "sees" (and hopefully a more accurate basis from which to determine new control parameters), but it also provides a basis from which considerable improvements can be made in a regression program to analyze on-line data.

Note that for the off-line analysis the input to the process is usually considered to be a physical variable in the system and, therefore, governed by Newton's laws of motion. A perfect pulse would be an ideal input to disturb the dynamics of a system; however it is not physically possible to instantaneously change any process variable, be it valve position, flow rate, temperature, concentration, etc. The input for off-line regression, therefore, has to generally be measured and recorded in addition to the output. However, the input to the
process for an on-line analysis is quite different. The input for on-line analysis is the output of a digital algorithm and is, therefore, nothing more than a program variable. The variable can in effect be changed instantaneously and a perfect pulse easily initiated. It is meaningless to sample the input in this case because it is a known function, free of both measurement and recording errors as well as process noise. The real advantage of this distinction is not clear at this point, but it will become apparent in the following development of an on-line regression program.

On-Line Regression

To pursue the development of an on-line regression program, consider the fit-error equation used by the off-line program (Equation 5.4). This equation can be expanded by squaring the terms in brackets to yield:

$$ E = SXX + 2SXU + SUU $$

(5.11)

or in terms of the minimization problem.

$$ E^* = \min_{\tau_1, \tau_2, \theta, \kappa} [SXX + 2SXU + SUU] $$

(5.12)

where the terms in the above equation are defined as follows:

$$ SXX = \Sigma x_1^2 + 2C_1 \Sigma x_{1-1} x_1 + 2C_2 \Sigma x_{1-2} x_1 + C_1^2 \Sigma x_{1-1}^2 $$

$$ + 2C_2 C_1 \Sigma x_{1-2} x_{1-1} + C_2^2 \Sigma x_{1-2}^2 $$

(5.13)
\[
SXU = B_1 \sum u_{i-n-1} x_i + B_2 \sum u_{i-n-2} x_i + B_3 \sum u_{i-n-3} x_i + C_1 B_1 \sum u_{i-n-1} x_{i-1} + C_2 B_2 \sum u_{i-n-2} x_{i-1} + C_3 B_3 \sum u_{i-n-3} x_{i-1} + C_2 B_2 \sum u_{i-n-2} x_{i-2} + C_3 B_3 \sum u_{i-n-3} x_{i-2} + C_2 B_2 \sum u_{i-n-2} x_{i-3} + C_3 B_3 \sum u_{i-n-3} x_{i-3}
\]

(5.14)

\[
SUU = B_1 \sum u_{i-n-1}^2 + 2B_2 B_1 \sum u_{i-n-2} u_{i-n-1} + 2B_3 B_1 \sum u_{i-n-3} u_{i-n-1} + B_2 \sum u_{i-n-2}^2 + 2B_2 B_3 \sum u_{i-n-3} u_{i-n-2} + B_3 \sum u_{i-n-3}^2
\]

(5.15)

As before the minimization indicated in Equation (5.12) can be implemented using a multidimensional search technique performed on Equation (5.11). Judging from the number of summation terms in Equations (5.13), (5.14), and (5.15), it might appear that instead of shortening the time required for each evaluation, it has instead been greatly increased. However, it will be shown that each summation term can be reduced to a constant which remains the same throughout the search, thereby greatly reducing the computation required for each iteration.

Immediately it is possible to reduce the summation terms contained in Equations (5.13) and (5.15) to constants. They are in no way dependent on any of the search parameters \((\tau_1, \tau_2, \theta, \text{ and } K)\), and need only be evaluated at the beginning of the search. The \(\Sigma x_i^2, \Sigma x_i x_{i-1}, \Sigma x_i x_{i-2}, \Sigma x_{i-1}^2\), etc. terms can easily be accumulated in the data collection program during the actual test period. Since \(u\) is a perfect
pulse, the summation terms involving only products of \( u \) can be determined by the following rather simple equations:

\[
\begin{align*}
\Sigma u_{i-n-1}^2 &= \Sigma u_{i-n-2}^2 = \Sigma u_{i-n-3}^2 = (N) \times H_p^2 \quad \text{(5.16 a)} \\
\Sigma u_{i-n-1}u_{i-n-2} &= \Sigma u_{i-n-2}u_{i-n-3} = (N-1) \times H_p^2 \quad \text{(5.16-b)} \\
\Sigma u_{i-n-1}u_{i-n-3} &= (N-2) \times H_p^2 \quad \text{(5.16-c)}
\end{align*}
\]

Where \( H_p \) = height of pulse

\( N \) = duration of pulse in terms of sampling period

The summation terms contained in Equation (5.14) cannot be reduced to constants quite as easily as the terms above. Note that all the terms in Equation (5.14) contain cross products of \( x \) and \( u \) (\( \Sigma u_{i-n-1}x_i \), \( \Sigma u_{i-n-2}x_i \), etc.). Unlike the summation terms containing only \( x \) or \( u \), the numerical value of summations involving cross-products depend on the value of \( n \) (determined by the dead time) which varies from iteration to iteration in the multidimensional search. However, the fact that \( u \) is a perfect pulse enables this apparent problem to be circumvented very nicely.

Consider the computation of the cross product term \( x_i u_{i-j} \) for the case when \( u \) is a perfect pulse as defined:

\[
\begin{align*}
u_{i-j} &= 0 \quad @ \quad (i-j) < 0 \quad \text{(5.17-a)} \\
\frac{H_p}{u_{i-j}} &= @ \quad 0 \leq (i-j) < N \quad \text{(5.17-b)} \\
u_{i-j} &= 0 \quad @ \quad N \leq (i-j) \quad \text{(5.17-c)}
\end{align*}
\]
(where \( u \) and \( x \) are both expressed as deviations about a reference point \( u_R, x_R \)).

The sum can be broken into intervals of constant \( u_{i-j} \) and expressed entirely in terms of the \( x \) data

\[
\sum_{i=1}^{M} x_i u_{i-j} = 0 \sum_{i=1}^{j} x_i + H^2 \sum_{i=j+1}^{j+N} x_i + 0 \sum_{i=j+N+1}^{M} x_i
\]

(5.18-a)

\[
\sum_{i=1}^{M} x_i u_{i-j} = H^2 \sum_{i=j+1}^{j+N} x_i
\]

(5.18-b)

The above equation can be further simplified by defining a vector \( \vec{S} \) such that each element is defined as follows:

\[
S_j = \sum_{i=1}^{M} x_i u_{i-j}
\]

(5.19)

for a pulse of height \( H_p \) and length \( N \)

\[
S_j = H^2 \sum_{i=j+1}^{j+N} x_i
\]

(5.20)

\[
S_j = S_{j-1} + H^2 (x_{j+N} - x_j)
\]

(5.21)

Note the vector \( \vec{S} \) can be very easily computed during the test period before the actual regression by the use to the equation above. Once the vector \( \vec{S} \) has been evaluated by the data collection program for all possible values of \( j \), none of the cross-product summation terms need to be evaluated by the regression program because all can be related to \( \vec{S} \) in the following manner:
At this point the development of a fast, efficient regression program is essentially complete. All the summation terms indicated in Equations (5.13), (5.14), and (5.15) have been reduced to constants thereby greatly reducing the time required for each iteration in the search. The need for lengthy data tables has also been eliminated, requiring instead a vector $S$ which is considerably shorter in length. In general, the form of the regression appears to be more in line with the requirements demanded by on-line applications.

General On-Line Approach

The approach demonstrated in the previous section for a second order lag plus dead time is perfectly general and can be applied to any model with a pulse transfer function of the form:

$$ x_i = \sum_{k=1}^{K_1-1} C_{k+1} x_{i-k} + \sum_{k=1}^{K_2} B_k u_{i-n-k} - S_{n-1} (5.23) $$

or in terms of the fit error

$$ E = \sum_{k=1}^{K_1} C_k x_{i-k} + \sum_{k=1}^{K_2} B_k u_{i-n-k} - S_{n-1} (5.24) $$
where

\[ K_1 = \text{number of } x \text{ terms in the pulse transfer function} \]
\[ K_2 = \text{number of } u \text{ terms in the pulse transfer function} \]
\[ C_k, B_k = \text{constants determined by the gain, delay, and time constants of the process model} \]

To summarize the general method, consider the computations and program requirements necessary to implement an on-line parameter regression for a process model described by the above equations (see Figure 5.2).

Data Collection Program: The data collection program plays a significant role in the overall strategy. It not only reads and records the data but simultaneously sums the appropriate terms and evaluates the \( S \) vector. This organization not only eliminates the need for data tables but also removes two fairly long summation terms from the actual regression program. The computation required each time a new data point is collected is as follows:

\[ S_{k-N} = S_{k-N-1} + H_p^2 (x_k - x_{k-N-1}) \] (5.25)

and

\[ \text{xx}_{i,j} = \text{xx}_{i,j} + x_i x_j \quad i = 1, 2, \ldots, K_1; \quad j = 1, 2, \ldots, K_2 \] (5.26)

Where

\[ k = \text{number of data points collected} \]
\[ N = \text{length of pulse} \]
\[ H_p = \text{height of pulse} \]
Figure 5.2. Micro-Flow Diagram: On-Line Identification
Regression Program: The regression program is organized as a main program, search routine, and a "fit-error" routine. The main program initializes the search and updates model parameters once the search is completed. The actual content of the main program will to a large extent depend on the search routine chosen and on the individual's own overall programming philosophy. However, one calculation which should logically be included in the main program is the following computation of the matrix \([UU]\)

\[
UU_{i,j} = N(l+j-1) \frac{H^2}{P} \quad i = 1, 2, \ldots, K_2; \quad j = 1, 2, \ldots, K_2 \quad (5.28)
\]

Note the above equation is the simplified version of the following expression taking advantage of the fact \(u\) is a perfect pulse.

\[
[SU] = \begin{bmatrix}
\Sigma u_{i-n-1}^2 & \Sigma u_{i-n-1}u_{i-n-2} & \ldots & u_{i-n-1}u_{i-n-K_2} \\
\Sigma u_{i-n-2}^2 & \ldots & \ldots & \ldots \\
\Sigma u_{i-n-K_2}^2 & \ldots & \ldots & \ldots \\
\end{bmatrix}
\]

(5.29)
The search routine conducts the search based on the minimization of the criterion function calculated by the "fit-error" routine. It does not require any programming peculiar to the on-line regression problem, and can be any one of many techniques currently available. However, care should be taken in selecting the technique to be used. Emphasis should be placed on core storage requirements and total running time. There are numerous programs which involve very sophisticated techniques; however, these programs generally require considerable room in core and because of the complex logic, require a comparatively large average iteration time. The simpler techniques are usually more effective in an on-line environment. Pattern Search (14), (17) is one such technique which has been found to be very effective.

The subroutine which evaluates the "fit-error" is the heart of the search. Based on the search parameters and on the process data (presented in terms of $X_i$, $U_i$ and $\bar{S}$) the program calculates a "fit-error" or criterion function which is used by the search routine to find the "best" parameters.

The first computation required is to evaluate the matrix $UX$ defined:

$$
[UX] = \begin{bmatrix}
\sum_{i=n-1}^{i=n-1-K_1} x_i \\
\sum_{i=n-2}^{i=n-1} x_i \\
\vdots \\
\sum_{i=n-K_2}^{i=n-K_2} x_i \\
\sum_{i=n-K_1}^{i=n-K_1} x_i \\
\vdots \\
\sum_{i=n-K_2}^{i=n-K_1} x_i \\
\sum_{i=n-K_1}^{i=n-K_1} x_i \\
\vdots \\
\sum_{i=n-K_1}^{i=n-K_1} x_i \\
\end{bmatrix}
$$

(5.30)
Recall the each element of \([UX]\) is a function of the dead time and can be computed each iteration in which the dead time has been changed as follows:

\[
UX_{i,j} = S_{n+i-j-1} \quad i = 1,2,\ldots - K_1; \quad j = 1,2,\ldots - K_2 \tag{5.31}
\]

The next step is to evaluate the constants in Equation (5.24) based on the trial process parameters. In general this is done in the following manner:

\[
C_i = f_i (K, \theta, \tau_1, \tau_2, \ldots) \quad i = 1,2,\ldots - K_1 \tag{5.32-a}
\]

\[
B_j = g_j (K, \theta, \tau_1, \tau_2, \ldots) \quad j = 1,2,\ldots - K_j \tag{5.32-b}
\]

The functions \(f_i\) and \(g_j\) depend entirely upon the process transfer function model (refer to equations which illustrate the equations which result from a second-order plug dead time model). In general, the necessary equations can be developed fairly easily from "modified" or "advanced" z-transforms (16).

The final step is the actual determination of the "fit-error". The necessary computations are as follows:

\[
SXX = \sum_{i=1}^{K_1} \sum_{j=1}^{K_1} C_i C_{j,XX} \tag{5.33-a}
\]

\[
SUU = \sum_{i=1}^{K_2} \sum_{j=1}^{K_2} B_i B_{j,UU} \tag{5.33-b}
\]

\[
SXU = \sum_{i=1}^{K_1} \sum_{j=1}^{K_2} C_i B_{j,UX} \tag{5.33-c}
\]
\[ E = SXX + 2SXU + SUU \]  

(5.34)

It should be noted that the calculated value of \( E \) is generally several orders of magnitude smaller than each individual term above. Therefore, it is advisable in most computers to use extended precision in the computation of Equations (5.33-a), (5.33-b), (5.33-c) and (5.34).

**On-Line Regression with Filtering:**

Since in many cases it is desirable to filter the data before parameter regression, consider the possibility of including this option in the on-line technique developed in the previous section. In conventional off-line analysis filtering can be easily accomplished by filtering not only the process output but also the process input. Provided the filter used on both the input and output data are identical, the regression analysis can be applied on the new data just as if the data were not filtered (4). However, in the on-line technique previously developed, this procedure cannot be realized. The process output data can be easily filtered; yet filtering the input data is not possible. The input is restricted by the development to be a perfect pulse and any filtering of \( U \) would destroy this pulse. On the surface this restriction appears to preclude filtering for on-line analysis; however, it is possible to circumvent the need to filter \( U \) by including the dynamics of the filter in the process model.

To illustrate the procedure consider the following regression problem. It is desired to fit a first order lag plus dead time model (Equation 5.35) to a set of on-line data.

\[ G(s) = \frac{Ke^{-\theta s}}{(1 + \tau_1s)} \]  

(5.35)
Due to process noise it has been found helpful to numerically condition the data before the regression by the following filter:

\[
H(s) = \frac{1}{(1 + \tau_F s)}
\]  

(5.36)

Combining the original model and the filter yields the following corrected model:

\[
HG(s) = \frac{ke}{(1 + \tau_1 s)(1 + \tau_F s)}
\]  

(5.37)

Using the pulse transfer function of the above model, the resulting minimization problem is as follows:

\[
E^* = \min_{\tau_1, \theta, K} C_1 x_i + C_2 x_{i-1} + C_3 x_{i-2} + B_1 u_{i-1-n-1}
\]

\[
+ B_2 u_{i-1-n-2} + B_3 u_{i-1-n-3}
\]  

(5.38)

Where the coefficients in the expression above \((C_1, C_2, C_3, B_1, B_2, B_3)\) are functions of the process parameters \(\tau_1, \theta, K\) as well as the filter constant \(\tau_F\). The equations necessary to evaluate the coefficients above are identical to the equation necessary to fit a second-order plus dead time model (see Equations 5.4-5.9); however, the filter constant \(\tau_F = \tau_2\) is not a search parameter and is, therefore, held constant throughout the search.

This same procedure can be applied to any process model and filter combination in which an analytical solution can be obtained. This is generally possible because low order models with dead time usually satisfactorily represent the process and a first-order filter provides
sufficient data conditioning in most cases (4).

Comparison of Convention and On-Line Regression

To quantitatively compare the conventional off-line regression to the technique developed in this article a study was conducted on an IBM 7040. Both techniques were programmed and used to analyze several sets of data for a first-order lag plus dead time model. To be realistic a first order filter was included in both programs. In every case the two methods converged to essentially the same answer and required about the same number of total functional evaluations. The on-line technique however showed considerable improvement in terms of running time. Using 500 data points the average running time per functional evaluation was about 0.70 sec for the conventional technique and about 0.043 sec for the on-line technique; an improvement of about 18/1. This improvement is even more exaggerated when more data points are used (35/1 @ 1,000 pts).

The savings in core storage by using the on-line technique is not as easily determined because it depends a lot on the individual computer and on how the data is stored but generally about 2-4 words of core are required for each data point pair (input - output) for a conventional program. Therefore, if 500 data points are to be analyzed, the conventional technique requires about 1000-2000 more words of core.

In Summary

This chapter demonstrates how significant improvements can be made in both program length and computation time for the nonlinear least-squares regression problem by restricting the process input to be a "perfect" square pulse. This restriction may be serious in
typical off-line applications where the input term is a physical variable and cannot be changed instantaneously; however, it conforms nicely with an on-line application where the input term is usually a variable in a digital program. The chapter also demonstrates how the technique can also be easily used to analyze data in which filtering is necessary.
NOMENCLATURE

K = Gain
θ = Dead time
τ₁, τ₂ = Time constants
E = Fit error
K* = Optimal gain
θ* = Optimal dead time
τ₁*, τ₂* = Optimal time constants
E* = Minimum fit-error
τ_F = Filter constant
xᵢ = Process output data
x̂ᵢ = Process model output data
uₐ = Process input data
T = Sample time
n = Integer number of sampling times in θ
N = Length of pulse
H_p = Height of pulse
M = Total number of data points
Cᵢ, Bᵢ = Coefficients of pulse transfer functions
[XX] = Matrix of Σxᵢxⱼ terms
[UU] = Matrix of Σuᵢuⱼ terms
[UX] = Matrix of Σuᵢxⱼ terms
S = Vector containing cross product summation terms
LITERATURE CITED


CHAPTER VI

FORMULATING THE LEAST-SQUARE REGRESSION FOR CONTINUOUS ANALYSIS

Introduction

With the advent of digital control an increased emphasis has been placed on such advanced control concepts as plant set-point optimization, non-interacting multivariable control, feedforward control, cascade control, etc. In order to successfully implement such concepts the control engineer is becoming more and more concerned with process modeling and process identification. This concern is complicated by the fact that many processes encountered in the process industries are non-stationary, that is, the "correct" or "best" model actually changes with time due to changes in such factors as operating level, catalyst age, environmental conditions, etc. Hence, if the full potential of the techniques mentioned above are to be realized, a fairly accurate process model must not only be determined but must be re-determined and up-dated as the changes in the process occur.

Numerous techniques have been developed to adapt to changes in process parameters, most of which require the system to be taken off control and disturbed so that data may be collected for either on-line or off-line analysis. These techniques in many cases cannot be justified even though adaptive modeling is needed. The requirement that the process be intentionally disturbed in most cases is undesirable and frequently completely impractical. Also, the actual computer programs required to perform the analysis are generally long both in
execution time and computer storage. This chapter will demonstrate that in most cases these requirements are not necessary for either on-line or off-line analysis and that the regression problem can be formulated so that changes in model parameters can be continually monitored under normal controlled operation without extensive use of valuable computer time or storage.

**Developing General Nonlinear Regression**

The general technique to be considered in this chapter is the least-square parameter regression. Many forms of this technique have been used successfully for updating model parameters (1) (2). The technique is very general and can be applied to most models commonly used to describe chemical processes. It is statistical by nature and particularly suitable for process data which is accompanied by noise.

Consider the development of a conventional least-square analysis using a first order lag plus dead time model (for simplicity the dead time is considered to be an integer multiple of the sampling time).

\[
G_m(s) = \frac{Ke^{-\theta s}}{(1 + \tau s)} \tag{6.1}
\]

**Process Parameters:**

- \(K\) = gain
- \(\tau\) = time constant
- \(\theta\) = dead time = \(nT\) (integer x sampling time)

The first step in developing a regression program is to define a criterion or measure of fit by which to judge which parameter values best describe the data. The criterion function used by all
"least-square" programs is defined as follows:

\[ E = \sum_{i=1}^{m} (x_i - \hat{x}_i)^2 \]  
(6.2)

where

\[ E \quad \text{fit error} \]

\[ x_i \quad \text{actual process output at point} \; i \]

\[ \hat{x}_i \quad \text{predicted process output at point} \; i \]

\[ m \quad \text{total number of data points} \]

This expression can be expanded by expressing \( \hat{x}_i \) in terms of the pulse transfer function corresponding to the dynamic model:

\[ \hat{x}_i = x_{i-1}e^{-T/\tau} + K(1-e^{-T/\tau})u_{i-n-1} \]  
(6.3)

or

\[ E = \sum \left( x_i - x_{i-1}e^{-T/\tau} - K(1-e^{-T/\tau})u_{i-n-1} \right)^2 \]  
(6.4)

where

\[ T \quad \text{sampling time} \]

The object of the regression is to find values of \( \tau_1 \), \( \theta \), and \( K \) which yield the minimum fit-error \( E^* \) subject to the constraint that \( \tau_1 \) and \( \theta \) are positives:

\[ E^* = \min_{\tau_1, K} \sum_{i=1}^{m} \left[ x_i - x_{i-1}e^{-T/\tau} - K(1-e^{-T/\tau})u_{i-n-1} \right]^2 \]  
(6.5)
The minimization indicated above can be easily implemented by performing a direct search of \( \tau_1, K, \) and \( n \) to find \( E^* \) by using one of the many multidimensional search techniques currently available (3), (4).

**Reducing Order of Search:**

In the example above it should be noted that the multidimensional search can be reduced to a much simpler one-dimensional search. Consider re-writing Equation (6.5) such that

\[
E^* = \min_{C_1, B_1, n} \sum (x_i - C_1 x_{i-1} - B_1 u_{i-1})^2 \quad (6.6-a)
\]

where

\[
C_1 = e^{-T/\tau} \quad (6.6-b)
\]

\[
C_2 = K(1 - e^{-T/\tau}) \quad (6.6-c)
\]

Note the search can be rearranged and separated into two parts

\[
E_n = \min_n \left[ E_n^* = \min_{C_1, B_1} \sum (x_i - C_1 x_{i-1} - B_1 u_{i-1})^2 \right] \quad (6.7)
\]

\( E_n^* \) in the above expression represents the results of an sub-optimization of the fit-error at a fixed value of dead time. Note \( E_n^* \) can actually be determined analytically since the equation is linear at constant dead time. Differentiating \( E_n \) with respect to both \( C_1 \) and \( B_1 \) and setting equal to zero results in two algebraic equations which can be solved for \( C_1 \) and \( B_1 \)

\[
\frac{\partial E_n}{\partial C_1} = 2 \left[ \sum_{i=1}^{m} x_i x_{i-1} - C_1 \sum_{i=1}^{m} x_{i-1}^2 - B_1 \sum_{i=1}^{m} x_{i-1} u_{i-1} \right] = 0
\]

(6.9)
Solving for $C_1$ and $B_1$ yields:

$$C_1 = \frac{\sum_{i=1}^{m} x_i x_{i-1} - C_1 \sum_{i=1}^{m} x_i x_{i-1}}{\sum_{i=1}^{m} x_i x_{i-1} - C_1 \sum_{i=1}^{m} x_i x_{i-1}}$$

$$B_1 = \frac{\sum_{i=1}^{m} x_i x_{i-1} - C_1 \sum_{i=1}^{m} x_i x_{i-1}}{\sum_{i=1}^{m} x_i x_{i-1}}$$

Using the above simplification the nonlinear regression now involves only a one-dimensional search for the optimal dead time ($n$). Each iteration in the search involves calculating the best $C_1$ and $B_1$ for that particular value of $n$ using Equations (6.10) and (6.11). The fit error corresponding to a particular value of dead time can be calculated from the expansion of the criterion function

$$E_n^* = \sum_{i=1}^{m} x_i^2 - 2C_1 \sum_{i=1}^{m} x_i x_{i-1} - 2B_1 \sum_{i=1}^{m} x_i u_{i-n-1}$$

$$+ \sum_{i=1}^{m} x_{i-1}^2 + 2B_1 C_1 \sum_{i=1}^{m} x_{i-1} u_{i-n-1}$$

$$+ B_1^2 \sum_{i=1}^{m} u_{i-n-1}$$

(6.12)
Once the dead time search on the above equation is complete the process parameters can be determined directly from the optimum search parameters by rearranging Equations (6.6-b) and (6.6-c) such that

\[ \tau_1^* = -\frac{T}{\ln (C_1^*)} \quad (6.13-a) \]
\[ K^* = \frac{B_1^*}{(1-C_1^*)} \quad (6.13-b) \]
\[ \Theta^* = (n^*)T \quad (6.13-c) \]

In general the above simplification, possible for the first order lag plus dead time, is also possible for any process where the number of zeroes of the transfer function is one less than the number of poles. Unfortunately this does not include one model which is commonly used in the chemical process industry, namely the second order lag plus dead time.

\[ G_m(s) = \frac{Ke^{-\Theta s}}{(1 + \tau_1 s)(1 + \tau_2 s)} \quad (6.14) \]

Consider the difficulty encountered when implementing a least square analysis of the second order lag.

Again restricting the dead time to be an integer multiple of the sampling time, the pulse transfer function and the resulting fit error equation are as follows:

\[ \hat{x}_i = C_1 x_{i-1} + C_2 x_{i-2} + B_1 u_{i-n-1} + B_2 u_{i-n-2} \quad (6.15) \]

or

\[ E = \sum_{i=1}^{n} \left[ \frac{x_i - C_1 x_{i-1} - C_2 x_{i-2} - B_1 u_{i-n-1} - B_2 u_{i-n-2}}{2} \right]^2 \quad (6.16) \]
\[ C_1 = e^{-\frac{T}{\tau_1}} + e^{-\frac{T}{\tau_2}} \]  
(6.17-a)

\[ C_2 = -\left(\frac{-T/\tau_1}{e^{-T/\tau_2}}\right) \]  
(6.17-b)

\[ B_1 = K \left[ \frac{(1-e^{-T/\tau_2})/\tau_1 - (1-e^{-T/\tau_1})/\tau_2} \right] \]  
(6.17-c)

\[ B_2 = K \left[ \left(\frac{1}{\tau_1} - \frac{1}{\tau_2}\right) e^{-\frac{T}{\tau_1 + \tau_2}} + e^{-\frac{T}{\tau_2}} - \left(e^{-\frac{T}{\tau_1}} / \tau_2\right) \right] \]  
(6.17-d)

It is possible to again formulate the regression such that the dead time is a separate search and also to solve analytically for the constants in the pulse transfer function. However in this case little can be gained from this procedure because the knowledge of \( C_1^* \), \( C_2^* \), \( B_1^* \), and \( B_2^* \) cannot be related back to the desired model parameters \( K^* \), \( \tau_1^* \), and \( \tau_2^* \). (Note rearranging Equations (6.17-a - 6.17-d) will not yield an unique set of equations defining \( K_1^* \), \( \tau_1^* \) and \( \tau_2^* \).)

On the surface it would appear that the only way to handle the second order model is to conduct a four dimensional search of Equations (7.17 and 7.16) for the optimum \( \tau_1 \), \( \tau_2 \), \( K \), and \( \eta \). However, with only a few assumptions it is possible to again simplify the problem and avoid the very time consuming multidimensional search.

Instead of using the pulse transfer function as a basis for the model it is possible for low values of sampling time to use a difference approximation such as the following:
\[
\left[ \frac{(x_i - x_{i-1})}{T} - \frac{(x_{i-1} - x_{i-2})}{T} \right] \frac{\tau_{1\tau_2}}{T} + \frac{(x_{i-1} - x_{i-2})(\tau_{1\tau_2})}{T} + x_{i-1} = Ku_{i-n-1}
\]

(6.18)

Which when solved for \(x_1\) yields the following model equation

\[
x_1 = C_1x_{i-1} + C_2x_{i-2} + B_1u_{i-n-1}
\]

(6.19-a)

\[
C_1 = -\frac{2\tau_{1\tau_2} + (\tau_1 + \tau_2)T + T^2}{\tau_{1\tau_2}}
\]

(6.19-b)

\[
C_2 = \frac{(\tau_1 + \tau_2)T + \tau_1\tau_2}{\tau_{1\tau_2}}
\]

(6.19-c)

\[
B_1 = \frac{KT^2}{\tau_{1\tau_2}}
\]

(6.19-d)

The above equations defining \(C_1\), \(C_2\) and \(B_1\) can for this formulation be rearranged so that once \(C_1^*, C_2^*\) and \(B_1^*\) are found analytically, the corresponding values of the process constants can be found.

**Program Requirements Due to Summation Terms:**

If the previous simplifications are applicable to the regression problem, considerable savings are realized in terms of execution time due to the reduction of the multidimensional search involving only one parameter. In general even the better multidimensional search techniques require a considerable number of function evaluations (i.e. the number of times the fit-error function is evaluated) before the optimum can be identified. The one dimensional search problem on the other hand requires much fewer function evaluations but is limited
Consider the computation involved when a single dimensional search formulation is possible. For each iteration in the search the only equations which must be evaluated are the coefficient equations (i.e., 6.10 and 6.11) and the fit error equation (i.e., 6.12).

Note, however, fundamental in each equation are the various product and cross-product summation terms \( \sum_{i=1}^{m} x_i x_i', \sum_{i=1}^{m} x_i x_{i-1}', \ldots \) which must be evaluated over the \( m \) data points collected for the regression. It is necessary to only evaluate the output product terms \( x_i x_{i-1}', x_i x_{i-2}' \ldots \) once, before the first iteration in the search; however, all other terms vary depending on dead time and must be re-evaluated each iteration. This not only means that the running time will be long especially if considerable data is taken (as is usually required in order to insure a good fit) but also means the entire data must be stored during the analysis resulting in a quite large core requirement.

**Eliminating Summation Terms:**

Continuing the development of the first order lag plus dead time regression consider the implications when each term in Equations (6.10), (6.11), and (6.12) are divided and multiplied by \( m \), the number of data points. Using the definition of the statistical mean the resulting equations can be expressed in terms of "average" values of the product and cross product terms such that
Where in terms of the statistical mean or arithmetic average

\[ \bar{x}_1 = \frac{1}{m} \sum_{i=1}^{m} x_1 \]

\[ \bar{x}_{i-1} = \frac{1}{m} \sum_{i=1}^{m} x_{i-1} \]

\[ u_{i-n-1}^{u_{i-n-1}} = \frac{1}{m} \sum_{i=1}^{m} u_{i-n-1}^{u_{i-n-1}} \]

Little of any actual simplification results from the above representation however consider the possibility of using an weighted estimate of the "average" of the product and cross product terms instead

\[
C_1 = \frac{x_i x_{i-1} u_{i-n-1} u_{i-n-1} - x_i u_{i-n-1} x_{i-1} u_{i-n-1}}{x_{i-1} x_{i-1} u_{i-n-1} u_{i-n-1} - 2 x_{i-1} u_{i-n-1} u_{i-n-1}} \quad (6.20)
\]

\[
B_1 = \frac{x_i x_{i-1} - C_1 x_{i-1} x_{i-1}}{x_{i-1} u_{i-n-1}} \quad (6.21)
\]

\[
E_m^* \frac{1}{m} E_n^* = \frac{x_i x_{i-1} - 2 C_1 x_{i-1} x_{i-1} - 2 B_1 x_{i-1} u_{i-n-1}}{x_{i-1} x_{i-1} + 2 B_1 C_1 x_{i-1} u_{i-n-1} + B_1^2 u_{i-n-1} u_{i-n-1}} \quad (6.22)
\]
of the true arithmetic average as indicated above (4).

The general definition of a weighted average of a time function over the interval $T_1$ to $T_2$ is as follows:

$$\bar{f}(t) = \frac{\int_{T_1}^{T_2} f(t)\phi(t)dt}{\int_{T_1}^{T_2} \phi(t)dt}$$  \hspace{1cm} (6.23)

Note if the weighting function, $\phi(t)$, is chosen to be the exponential $e^{-\alpha t}$ and $T_1$ is selected to be $-\infty$, the above equation reduces to the following:

$$\bar{f}(T) = \alpha e^{-\alpha T} \int_{-\infty}^{T} f(t) e^{\alpha t}dt$$  \hspace{1cm} (6.23-a)

or

$$\bar{f}(t) = \alpha \int_{-\infty}^{T} f(t)e^{-\alpha(T-t)}dt$$  \hspace{1cm} (6.23-b)

This expression indicates that the exponentially weighted estimated of $f(t)$ is nothing more than the output of a first order filter with a filter constant of $1/\alpha$ and can be easily implemented numerically to estimate the necessary terms in Equations (6.20), (6.21), and (6.22).

**Advantage of Weighted Average Formulation**

Figure 6.1 illustrates a flow diagram of one possible means of implementing the first order lag plus dead time regression using the weighted average. Note that in terms of the program requirements it is a big improvement over conventional least square formulations. In
Data Collection and Filtering

\[ J = N_{\text{MAX}} + 2 \]

\[ J = J - 1 \]

\[ U(J) = U(J - 1) \]
\[ UU(J) = UU(J - 1) \]

\[ IS \]
\[ J = J_{\text{MIN}}? \]

\[ XX11 = XX00 \]
\[ U(1) = U0 \]

\[ UU(1) = UU(1) * F1 + U0 * U0 * F2 \]
\[ XX00 = XX00 * F1 + X0 * X0 * F2 \]
\[ XX01 = XX01 * F1 + X0 * X1 * F2 \]
\[ D0201 = I_{\text{MIN}}, I_{\text{MAX}} \]
\[ XO(I) = XO(I) * F1 + X0 * U(I) * F2 \]
\[ X1U(I) = X1U(I) * F1 + X1 * U(I) * F2 \]

\[ X1 = XO \]

RETURN

Nonlinear Regression

One-Dimensional Search

Logic to Vary \( N \)

\[ A = (UU(N) * XX01 - XO(U(N)) * X1U(N)) / (UU(N) * XX11 - X1U(N) * X1U(N) * 2) \]
\[ B = (XO(U(N)) - A * X1U(N)) / UU(N) \]

\[ E = XX00 - 2 * AX01 - 2 * B * XO(U(N)) + A * A * XX11 + 2 * A * B * X1U(N) + B * B * UU(N) \]

\[ IS \]
\[ N \text{ Optimum?} \]

\[ TM = -TS / \text{ALOG}(A) \]
\[ KM = B / (1 - A) \]
\[ DELM = (N - 1) * TS \]

RETURN

Figure 6.1 Simplified Flow Diagram of "Weighted-Average" On-Line Least-Square Analysis of Process Data for a First Order Lag Plus Dead Time Model.
Figure 6.1 the need for long data tables characteristic of conventional programs have been eliminated. Instead a relatively short set of tables are used in which to store a short record of the control action, $U(N)$, and the weighted average of the cross product and product terms involving the control action, $UU(N)$, $XOU(N)$, $X1U(N)$. The length of these tables need only be sufficient to include the range of expected dead time.

The running time of the weighted average formulation is also a significant improvement over the conventional formulation. Note the regression equation in Figure 6.1 is free of the long summation terms and can be executed quite fast compared to conventional analysis.

The weighted average formulation also requires no specific test period as in conventional regression. The data can be filtered continuously and either periodically or continuously analysed for new parameters without the need to initialize or re-start the data collection.

Another advantage to the "weighted average" formulation is that when integer arithmetic is used (as is typically the case in process control application) the numerical problem of integer overflow does not exist like it does when the terms are actually summed.

Performance of the Weighted Average Formulation:

To illustrate the performance of the regression when the weighted average simplification is used the following first order lag plus dead time process was simulated and studied by a digital simulation.

$$\frac{-0.5s}{1+ns}$$

$\quad G_p(s) = \frac{e}{(1+s)} $  \hspace{1cm} (6.24)
Figure 6.2. Performance of the "Continuous" Least-Square Regression to a Closed Loop Change in Set-Point \( G_p(s) = e^{-0.5s}/(1+s) \).
Figure 6.3. Performance of "Continuous" Least-Square Regression to Pure Noise \( G_p s = e^{-5s/(1+s)} \).
Figure 6.4-a. Dynamic Response of the "Continuous" Least-Square Regression to a Step Change in Process Time Constant.
Figure 6.4-b. Dynamic Response of the "Continuous" Least-Square Regression to a Step Change in Process Gain.
Figure 6.4-c. Dynamic Response of the "Continuous" Least-Square Regression to a Step Change in Process Dead Time.
In every case investigated a sampling time of 0.1 and a filter constant of 5.0 were used.

The results of the first study is illustrated in Figure 6.2. In this study the continuous response of the regression to a closed loop step change in set-point. Note from Figure 6.2 the analysis is quite satisfactory and yields the correct value of gain, time constant, and dead time throughout the entire response. This ability to operate satisfactorily under closed loop operation is an important improvement over conventional on-line regression programs which normally require the controller to be taken off-line and the system manually disturbed with a step or pulse input.

Figure 6.3 illustrates the behavior of the "continuous" regression to data generated by an input consisting of pure noise. Note the regression actually performs quite well under these extreme conditions. The gain tends to drift in time; however, the dead time and time constant parameters are quite good throughout the response.

To determine how "fast" the regression responds to changes in process parameters the digital simulation was again studied with a pure noise input. At time one, simulated values of time constant, gain, and dead time were changed in three separate runs illustrated in Figure 6.4-a, 6.4-b, and 6.5-c. Note in each case the regression does not exactly follow the parameter values but does over a period of time adjust very well considering the input output data which is used to make the analysis.

**Example - Non-Ideal Plug Flow Reactor**

To illustrate the performance of the continuous least-square regression on a practical example the hypothetical reactor illustrated
in Figure 6.5 was simulated in detailed on the digital computer. The system consists of a non ideal plug flow reactor in which the following second order reaction is taking place

\[ k_1(T) \]

\[ A + B \xrightarrow{k_1(T)} R \]

Flowing into the reactor is a recycle stream enriched with the reactant B at the rate of \(20 \text{ ft}^3/\text{min}\). The reaction is controlled by a stream of pure A which can be varied between \(0.0\) and \(12 \text{ ft}^3/\text{min}\).

Figure 6.6 illustrates how the gain, time constant, and dead time of the reactor model vary greatly over the range of possible control actions. These parameter profiles were calculated using the continuous least-square regression program developed in this chapter. The only modification which was necessary in this application was to limit the change in each calculated value of the parameter so that the sharp changes which occurred with each change in operating level would be damped out. These sharp changes are the result of the fact the reactor is much more complex than the simple first order lag plus dead time model assumed in the regression. Note from Figure 6.6 that the initial response of the reactor to an increase in the control variable is to actually decrease the conversion for a short period of time due to the initial cooling effect of an increase in flow rate. This fact alone is enough to temporarily upset the calculations. All in all, however, the regression seems to indicate the general trend of model parameter which would be anticipated based on the step response at the various operating levels.
Figure 6.5. Non-Ideal Plug Flow Reactor Under Computer Control.
Figure 6.6. Model Profile at Various Operating Levels in a Non-Ideal Plug Flow Reactor Using Continuous Least Square Analysis.
In Summary this chapter demonstrates that for most common models the nonlinear least-square parameter regression can be formulated to avoid the need for more than a single dimensional search. Also presented is a method for formulating the nonlinear least square regression so that an actual "continuous" on-line analysis is possible and can be applied to both closed loop data and to data consisting of pure noise.
NOMENCLATURE

K = Gain
θ = Dead time
τ₁, τ₂ = Time constants
E = Fit error
K* = Optimal gain
θ* = Optimal dead time
τ₁*, τ₂* = Optimal time constants
E* = Minimum fit-error
x₁ = Process output data
xᵢ = Process model output data
uᵢ = Process input data
T = Sample time
n = Integer number of sampling times in θ
m = Total number of data points
Cᵢ, Bᵢ = Coefficients of pulse transfer functions
Cᵢ*, Bᵢ* = Optimum coefficients of pulse transfer function
f(t) = Arbitrary time function
Ø = General weighting function
α = Pole of weighted average filter
**FLOW DIAGRAM KEY**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XO</td>
<td>Process output sampled at time zero.</td>
</tr>
<tr>
<td>UO</td>
<td>Manipulated variable calculated at time zero.</td>
</tr>
<tr>
<td>U(N)</td>
<td>Table of past values of the manipulated variable</td>
</tr>
<tr>
<td>UU(N)</td>
<td>Table containing the weighted average of $u_{i-n-1}$ for various values of the dead time $n$.</td>
</tr>
<tr>
<td>XOU(N)</td>
<td>Table containing the weighted average of $x_i u_{i-n-1}$ for various values of the dead time $n$.</td>
</tr>
<tr>
<td>XIU(N)</td>
<td>Table containing the weighted average of $x_{i-1} u_{i-n-1}$ for various values of the dead time $n$.</td>
</tr>
<tr>
<td>XX00</td>
<td>$\bar{x}_1 x_1$</td>
</tr>
<tr>
<td>XX01</td>
<td>$\bar{x}<em>1 x</em>{i-1}$</td>
</tr>
<tr>
<td>XX11</td>
<td>$\frac{x_{i-1} x_{i-1}}{x_{i-1} x_{i-1}}$</td>
</tr>
<tr>
<td>A1B</td>
<td>Coefficients of the pulse transfer function</td>
</tr>
<tr>
<td>E</td>
<td>Fit error</td>
</tr>
<tr>
<td>TM</td>
<td>Time constant determined by the regression</td>
</tr>
<tr>
<td>KM</td>
<td>Gain determined by the regression</td>
</tr>
<tr>
<td>DELM</td>
<td>Delay determined by the regression</td>
</tr>
</tbody>
</table>
LITERATURE CITED


CHAPTER VII

PREDICTOR ALGORITHMS FOR DIRECT DIGITAL CONTROL

Introduction:

In recent years the high speed digital computer has made an impact in the area of process control. Many sophisticated computer control concepts and systems have been designed to utilize the tremendous capability and the flexibility of the digital computer. Yet surprisingly, only a little of this effort has been actually directed to the first level of control where the process variable is directly measured and directly controlled. Most have been concerned with the second level of control including concepts such as set-point optimization (i.e. supervisory control), multivariable noninteracting control, adaptive control, cascade control, feedforward control, etc.

This chapter will investigate the first level of control and present a general technique for improving "old reliable" conventional control algorithms for feedback compensation with a digital computer.

Paradox of Digital Computer Control

Consider the following question: What is to be gained if anything by simply replacing the conventional analog controller, such as the proportional (P), proportional-integral (PI), and proportional-integral-derivative (PID), with a corresponding digital controller? Figure 7.1 is the result of a study of a three-mode (PID) digital algorithm controlling the second-order process.
Figure 7.1. Effect of Sampling Time on System Performance (1).

\[ G_p(s) = \frac{1.0}{(1+\tau_1 s)(1+\tau_2 s)} \]
At various combinations of sampling time $T$ and process time constant $\tau$, the optimum process response to a change in setpoint was determined by tuning the controller parameters to minimize $\text{ISE}$ (integral of square error). Note from Figure 7.1 if one were to choose the optimum sampling time the obvious result would be $T = 0.0$ or continuous sampling. For each value of time constant the system performance deteriorates ($\text{ISE}$ increases) as the sampling time becomes larger.

This would indicate an apparent paradox in the application of direct digital control (DDC). One would expect that, because of the tremendous expense and effort involved with a digital control system, rather significant improvements in process performance would be necessary in order to justify the digital computer. However, from Figure 7.1 it is obvious that instead of improving the process response the digital computer actually penalizes the system when conventional algorithms are used. The answer to the paradox is of course that the computer system must be justified by improved overall plant performance or on the basis of the need for some control strategy which would be difficult or expensive to implement using analog equipment. A computer system cannot be justified simply on the basis of replacing a conventional feedback analog control scheme. However, one partial answer to this problem of justification is to break free of the conventional algorithms and use instead algorithms specifically designed for digital computer control.

$$G_p(s) = \frac{1}{(1 + \tau s)(1 + s)} \quad (7.1)$$
Dynamics of the Digital Interface:

In order to understand why conventional control deteriorates when implemented in a sampled data environment, consider the difference between an analog loop and the corresponding digital loop. The basic physical difference in the two systems is the interface required to connect the discrete digital computer to a continuous process. If the same algorithm is used in both systems the difference is essentially the dynamic contribution of the digital interface, which in most process applications is much the same as a pure dead time or transport lag equal to one half the sampling time. Therefore, the digital algorithm must not only control the original process but also an additional dead time. Since conventional algorithms are all very sensitive to dead time, it is understandable why digital control produces poorer system performance.

Compensation for Interface Dynamics - Simple Case

In developing a feedback control algorithm which does not penalize the sampled data (discrete) control system so severely as do the conventional algorithms, it is helpful to consider the simplification that the digital interface acts much like a pure dead time. Using this fact it is possible to eliminate the effect of the interface using a conventional control loop with an "analytical predictor" in the feedback as illustrated in Figure 7.2. The analytical predictor attempts to eliminate the effective dead time of the interface by analytically predicting the value of the process output at one-half sampling time in the future. Corrective action is based on this predicted value instead of on the actual value.
Analytical Predictor Scheme

Figure 7.2. Compensating for Digital Interface Using Analytical Predictor Scheme for Eliminating Dead Time.
To illustrate the analytical predictor concept consider the simple case of controlling a process described by a pure first-order lag:

\[ x(t) + \tau \frac{dx(t)}{dt} = K u(t) \]  

(7.2)

Using a simple proportional controller, the conventional control law for this system, based on the process output sampled at time zero \( x_o \), is:

\[ u_o = K_c (R_o - x_o) \]  

(7.3)

To eliminate the effective dead time (equals \( T/2 \)) of the interface, the following control law would be necessary:

\[ u_o = K_c (R_o - x_{\frac{T}{2}}) \]  

(7.4)

Where \( x_{\frac{T}{2}} \) is the value the process output will reach one-half sampling time in the future. Although it is not available at time zero, a good approximation of \( x_{\frac{T}{2}} \) can be predicted from the analytical solution of the system Equation (7.2).

\[ x_{\frac{T}{2}} = \left( 1 - e^{-T/2\tau} \right) \Delta u_o + x_o e^{-T/2\tau} \]  

(7.5)

Equation (7.4) and (7.5) can be solved simultaneously to eliminate \( x_{\frac{T}{2}} \), yielding an expression for \( u_o \) which contains both the predictor and the proportional modes combined to form a complete digital algorithm:
This section will consider the development of a predictor-controller based on a first-order lag plus dead time model, which is much more representative of the typical processes found in the chemical industry:

\[ G(s) = \frac{G(s)}{(1 + \tau s)} e^{-\theta s} \]  

(7.7)

Although this model in many cases may be a drastic oversimplification, most processes yield a highly overdamped response which can be approximated by this model (2).

To illustrate the basic difference in the development of the predictor algorithm when the process contains a dead time, consider the process data in Figure 7.3. Illustrated is an arbitrary sequence of control actions and the corresponding response of a first-order system. Note the effect of the process dead time is to shift the sample output back in time so that the actual value of the process sampled at time zero is \( x_0 \) instead of \( x_0 \) (which would have been sampled if the process contained no dead time). The predictor therefore must not only predict over one-half the sampling time but must also predict over the dead time of the process. From Figure 7.3 at time zero the algorithm is expected to calculate the control \( u_o \) on the basis of \( x_{\hat{z} + \theta + N} \).

\[ u_o = K_c \left( R_o - x_{\hat{z} + \theta + N} \right) \]  

(7.8)
Figure 7.3. Effect of Dead Time on Sampled Process Information.
where \( \beta \) = fractional sampling time delay

\( N \) = integer sampling time delay

\( \theta = (N+\beta)T \) = total process delay

In the prediction of \( x_{2+\beta+N} \), the system equations can be solved analytically over the necessary sampling time to yield the following equation:

\[
\begin{align*}
x_{2+\beta+N} &= (1-A)Gu_o + A(1-B)G \sum_{i=1}^{N} B^i u_i + AB^N G(1-C)u_{N-1} + Cx_o \\
&= -(1-A)Gu_o + A(1-B)G \sum_{i=1}^{N} B^i u_i + AB^N G(1-C)u_{N-1} + Cx_o \\
&= -(1-A)Gu_o + A(1-B)G \sum_{i=1}^{N} B^i u_i + AB^N G(1-C)u_{N-1} + Cx_o
\end{align*}
\]

(7.9)

where:

\( A = e^{-T/2r} \)

\( B = e^{-T/r} \)

\( C = e^{-\beta T/r} \)

As before, Equation (7.8) and (7.9) can be solved simultaneously to eliminate \( x_{2+\beta+N} \) to give the following algorithm:

\[
\begin{align*}
u_o &= \frac{K_c}{1 + K_c(l-A)} \left[ R_o - A(l-B)G \sum_{i=1}^{N} B^i u_i - AB^N G(1-C)u_{N-1} + x_o \right] \\
&= \frac{K_c G + 1}{K_c G} \frac{1}{SP_o} \quad \text{(necessary to force the closed loop gain to be unity)} \\
&= \frac{K_c G + 1}{K_c G} \frac{1}{SP_o} \quad \text{(necessary to force the closed loop gain to be unity)}
\end{align*}
\]

(7.10-a)

(7.10-b)

The algorithm is of the general form:

\[
\begin{align*}
u_o &= C_1 SP_o - C_2 x_o - C_3 u_{N-1} - C_4 \sum_{i=1}^{N} B^i u_i \\
&= C_1 SP_o - C_2 x_o - C_3 u_{N-1} - C_4 \sum_{i=1}^{N} B^i u_i
\end{align*}
\]

(7.10-c)
Designing for Load Disturbances

Consider the development of a predictor algorithm to take into consideration the possibility that load disturbances enter the process loop, as shown in Figure 7.2, as a step input of unit height.

Recognizing that a load change will also force the system the control law becomes

\[ u_0 + z_o = K_c R_o - x_{2N} \]  \hspace{1cm} (7.11)

the analytical prediction of \( x_{2N} \) becomes

\[ x_{2N} = G(1-A)(u_0 + z_o) + A(1-B)G \sum_{i=1}^{N} B^{i-1}(u_1 + z_1) + G(1-C)(u_{N-1} + z_{N-1}) + Cx_o \]  \hspace{1cm} (7.12)

Solving Equation (7.11) and (7.12) yields an algorithm of the form:

\[ u_0 = C_1 sP_0 - C_2 x_o - C_3(u_{N-1} + z_{N-1}) - C_4 \sum_{i=1}^{N} B^{i-1}(u_1 + z_1) - z_o \]  \hspace{1cm} (7.13)

For the many applications where the load changes can be directly measured and recorded the above algorithm is ideal. However in reality even in such cases where all the major load changes can be measured, it is probable that other unmeasurable disturbances terms will enter the loop, affecting the system performance. Consider modifying the predictor algorithm to take into consideration such unmeasured disturbances.

At time zero it is possible to determine if an additional
disturbance has entered the system by comparing the actual sampled process output, $x_o$, with a predicted output, $\hat{x}_o$. When the two differ significantly this can be assumed to be an indication that an additional disturbance, not considered in predicting $x_o$, has entered the loop. This reasoning can be used very nicely in the construction of an "integral" type mode for the predictor algorithm to eliminate offset due to such unknown disturbances.

The integral mode consists of numerically integrating the difference between the actual output and the predicted output in the following manner:

$$\hat{x}_o = Bx_1 + G(1-B/C)(u_{N-1} + z_{N-1} + d_1) - B(1-1/C)(u_N + z_N + d_1)$$

(7.14)

$$d_o = d_1 + K_r(x_o - \hat{x}_o)T$$

(7.15)

The predictor algorithm (Equation 7.13) must be modified only slightly to include this calculated disturbance term

$$u_o = C_1SP_o - C_2x_o - C_3(u_{N-1} + z_{N-1} + d_o) - C_4 \sum_{i=1}^{N} B^i(u_i + z_i + d_o)$$

$$- (z_o + d_o)$$

(7.16)

Note the "integral" mode described above is very similar to the integral or reset mode used in conventional algorithms. However, one basic difference exists which is important in terms of system performance. The conventional integral mode is based on the integral of the difference between the set-point and the sampled output and is
therefore always "active" as long as an error exists. On the other hand the integral mode described above is "active" only when an additional disturbance enters the loop and, therefore, does not interact with the proportional mode. This "separation" of the modes becomes important when considering the response of a system to both set-point changes and load changes as will be demonstrated when the actual performance of the predictor algorithm is considered.

Tuning the Predictor Algorithm

In general specifying the control parameters for a conventional one-, two-, or three-mode algorithm is quite a problem especially in a sampled data environment. However, for the proportional mode of the predictor algorithm "tuning" can be done quite simply. Consider the following criterion for good control. Since the process dynamics are assumed to be first order and since, for the moment, physical constraints are not being considered, it is possible to require the controller to force the system to respond to a change in set-point in one sampling time without overshoot or oscillation. In general, this criteria is not possible to implement using conventional algorithms except for the case where the process dead time is zero. However, it is possible to realize using the proportional-predictor. When the process model is accurate, the loop (regardless of the magnitude of dead time) behaves like a loop containing no dead time.

Concentrating on the case where the dead time is zero, consider the solution of Equation (7.2) from time zero to time T.

\[ x_1 = G(1-B)u_0 + Bx_0 \]  

(7.17)
where \( u_o \) is described by the proportional-predictor algorithm

\[
\begin{align*}
\frac{K_c}{1 + K_c G(1-A)} & \left[ \frac{(K_c G + 1)}{K_c G} S_p - x_o A \right] \\
\end{align*}
\]

(7.18)

Using the tuning criteria above requires

\[
x_1 = S_p
\]

(7.19)

The three equations above can be solved simultaneously and after a little algebraic manipulation will yield the following expression

\[
K_c = \frac{A}{G(1-A)} = \frac{e^{-T/2\tau}}{G(1-e^{-T/2\tau})}
\]

(7.20)

The integral predictor mode can also be tuned in similar manner. Requiring the system to reach the desired set-point in one sampling time, it becomes necessary for the integral mode to react immediately to "eliminate" the disturbance from the process loop so that the proportional mode can bring the system back as desired. The value of \( K_c \) required to calculate the disturbance in one sampling time can be found by again considering the case of no dead time.

Suppose a first order process is operating under steady-state conditions at the desired set-point when a step disturbance, \( q \), enters the loop at time \(-T\) undetected, so that over one sampling period the system responds in the following manner:

\[
x_o = G(1-B)(u_{-1} + q) + Bx_{-1}
\]

(7.21)

Since at time \(-T\) the disturbance was calculated to be zero \( d_{-1} = 0.0 \)
the process value predicted by the controller (Equation 7.14) is as follows:

$$\hat{x}_o = G(1-B)(u_1 + 0.0) + Bx_1$$

(7.22)

Substituting the above equations into the "integral" equation results in the following prediction of the disturbance

$$d_o = d_1 + K_r (x_o - \hat{x}_o) + 0.0 + K_r G(1-B)q T$$

(7.23)

Since the tuning criterion requires that the calculated disturbance be equal to the actual disturbance the substitution $d_o = q$ can be made and the resulting expression solved for $K_r$.

$$K_r = \frac{1/T}{G(1-B)} = \frac{1/T}{G(1-e^{-T/\tau})}$$

(7.24)

**Performance of the Predictor-Algorithm -- Ideal Conditions**

Before proceeding to the "real world" consider the performance of the predictor-algorithm when ideal process modeling is possible. For illustration consider the following process:

$$G_p(s) = \frac{1.0}{(1+1.0s)^{-0.5s}} e^{-0.5s}$$

(7.25)

controlled by a proportional-predictor algorithm with a sampling time of $T = 0.1$.

**Set-Point Response:** Figure 7.4-a illustrates the response of the predictor-algorithm (P-A) to a step change in set-point and compares the response of the conventional proportional (P) and proportional plus integral controllers (PI). (It should be noted that the reason no "off-set" as such is observed in the two proportional algorithms
Figure 4-a. Performance of the Predictor Algorithm to a Change in Set-Point -- Perfect Modeling.

Figure 4-b. Performance of the Predictor Algorithm to a Change in Load -- Perfect Modeling.
(P and P-A) is because in both, the set-point was calibrated by Equation (7.10-b) so that the closed loop gain would be 1.0.) The P-A algorithm responds in one sampling time with no overshoot and no oscillation (as specified by the gain calculated using Equation 7.18). The conventional algorithms on the other hand, yield both overshoot and oscillation and result in a large settling time (20 to 40 times as large as that required by the P-A controller).

**Disturbance Response:** Figure 7.4-b illustrates the performance of the three algorithms to a step change in load. Three cases are included for the predictor-algorithm, one for each possible way of handling disturbances entering the system:

- **Case 1** - disturbance ignored in the P-A algorithm (Equation 7.10)
- **Case 2** - disturbance measured and fed forward to the P-A algorithm (Equation 7.13)
- **Case 3** - disturbance calculated from feedback data alone (Equations 7.14, 7.15, and 7.16)

Note, in Figure 7.4-b the conventional algorithms do not compare even to the worst case of the P-A algorithm. Both PI and P require considerable time to restore the system to steady-state operation. As would be expected the PI controller is able to finally obtain the desired set-point. However, the P controller reaches instead a steady-state with a rather large value of "off-set".

In comparing the different P-A cases as would be expected the best response occurs when the disturbance is actually measured and fed forward (Case 2). Only a slight "bump" is observed at TIME = 0.6. This occurred because the disturbance entered the system immediately after one sampling time and could not be corrected until the next sampling
time. This represents the worst possible performance. If the disturbance had entered the system just before a sampling time the disturbance could have been corrected before it upset the system.

The next best method is Case 3 where the disturbance was calculated from the process data by the integral-predictor mode. Note that the effect of the disturbance is much greater than for Case 2. This is because the disturbance must go completely through the process and dead time before the calculation can be made and corrective action taken. Once the disturbance has been calculated and effectively removed from the process loop the system requires only one sampling time to respond and return without overshoot or oscillation to the desired set-point.

As would be expected the worst of the three P-A methods is the case where the disturbance is completely ignored in the algorithm (Case 1). Note for this case the control calculates the control action necessary to return the system to the set-point if no disturbance were in the loop and is therefore only sufficient to stop the drift. This action results in off-set which is a function of the dead time in the system.

Performance of the Predictor-Algorithm -- "Real World" Conditions

In the previous section the Predictor-Algorithm was shown to be quite effective under ideal conditions. Consider now the performance of the algorithm in the "real world" where it is virtually never possible to obtain the ideal situation assumed in the previous section.

Physical Constraints: One of the first problems with implementing the predictor-algorithm in the real world would seem to be physical constraints on the allowable control action. To demand that the
system respond to a change in set-point in one sampling time is fine; however in many (and perhaps most) cases this demand is physically impossible. In general, there are always physical limits on how far a valve can be opened or closed, there are limits on the rate at which heat can be added to or taken away from a system, etc. To illustrate the effect of the constraints on the predictor-algorithm, the magnitude of the allowable control action for the system described by Equation (7.24) was constrained to be less than or equal to 1.5. The response of the various controllers to a unit change in set-point is shown in Figure 7.5. Because of the constraints the P-A algorithm is not able to reach the new set-point in one sample time; however, it does respond as fast as physically possible and as before does not overshoot or oscillate about the set-point. It essentially performs as a bang-bang controller when necessary. This is a significant improvement over conventional methods which require two separate modes, one to handle small disturbances which result in relatively small corrective action and one to handle large disturbances (such as set-point changes) which result in saturation. The P-A algorithm handles both cases easily with no special modes or other requirements such as non-linear switching curves. The only requirement is that the "actual" value that the manipulated variable was able to obtain be used in the algorithm instead of the calculated values. In cases where it is practical the best way to accomplish this requirement is to feed back the control data as well as the process data. In cases where this is not possible good results will be obtained by actually putting saturation in the controller.
Constraints:

\[ |u_0| \leq 1.5 \]

Model:

\[ G_p(s) = \frac{1.0}{0.6s} \quad \text{e}^{0.6s} \]

Sampling Time = 0.1

Figure 7.5. Effect of Hard Constraints on the Predictor-Algorithm.
Modeling Errors: Even though a first-order lag plus dead time is representative of most processes found in the chemical industry, in reality an actual pure first order lag plus dead time is rare. The process will probably be either a very high order system or a distributes parameter system. To illustrate the effect that modeling error of this type has on the performance of the loop consider the following high order process.

\[
G_p(s) = \frac{1.0}{(1+0.5s)^{10}} \tag{7.28}
\]

The above process can be represented by the following first order lag plus dead time (based on a "least-square" fit):

\[
G_n(s) = \frac{1.0}{(1+2.2s)^e}^{-3.24}
\]

The dead time term in the model is quite large but the actual process contains no dead time.

Figure 7.6 illustrate the response of the system to a change in set-point and to a change in load. As shown in Figure 7.6-a, for a change in set-point the P-A algorithm is not able to respond in one sample time as in the case of perfect modeling; however, it does show considerable improvement over the conventional algorithms demonstrating a much faster initial rise and a shorter settling time.

For a load change, as before, the performance of the predictor algorithm depends on the method of handling the load change. When the load is completely ignored (Case 1) the system results in off-set; however, when the load is included (Cases 2 and 3) the off-set actually measured and fed forward (Case 2) only a slight up-set is observed;
Figure 7.6-a Effect of Severe Modeling Errors on the Predictor-Algorithm -- Set-Point Change.
Figure 7.6-b. Effect of Severe Modeling Errors on the Predictor-Algorithm -- Load Change.
yet, when the disturbance is removed by the integral-predictor mode the system is up-set but responds very rapidly to eliminate the offset.

The action of the various controllers under these two conditions illustrate the significance of "separation" of the proportional and the integral modes. Due to the separation of the modes, the predictor-algorithm performs the same to a change in set-point regardless whether the integral-predictor mode is included (Case 3) or not included (Case 1). But, for the conventional controller, the addition of an integral mode (PI) actually penalizes the performance of a pure proportional controller (P). This is due to the interaction of the conventional integral and reset modes. The "off-set" which results from a set-point change can be eliminated immediately in the proportional controller by calibration but must be taken out with time in the PI controller.

The effect of severe modeling errors on tuning parameters should be mentioned. In this problem, in spite of the modeling errors, the proportional-predictor mode could be tuned quite well by using the analytical expression shown in Equation (7.20). In fact, the proportional-predictor mode responded well and was quite stable regardless the gain used, including values of $K_c$ approaching infinity. The integral mode, on the other hand, was very sensitive. It was found that due to the severe modeling errors Equation (7.24) yielded values of $K_r$ which resulted in an unstable response to both a change in set-point and to a change in load. For this problem it was necessary to use a value of $K_r$ one-tenth the value predicted assuming perfect modeling.
Example - Non Ideal Plug Flow Reactor

To illustrate the performance of the predictor algorithm on a practical example the hypothetical reactor illustrated in Figure 7.6 was simulated in detail on a digital computer and controlled using both the P-A algorithm developed in this chapter and the PI algorithm conventionally used.

The process consists of a jacketed, non-ideal plug flow reactor in which the following second order reaction occurs

$$A + B \xrightarrow{k_1} R \quad k_1 = 175 e^{-500/T}$$

The reactor is fed at the rate of 18 ft$^3$/min by a recycle stream enriched with reactant B and controlled by manipulating a stream of pure A which can be varied between 0.0 and 12.0 ft$^3$/min.

In order to represent the reactor as a first order lag plus dead time as required in the development of the predictor algorithm a process reaction curve was obtained at the desired operating level and used to determine values of gain time constant and dead time (3). These values were used to both implement the predictor algorithm and to tune the PI controller (2).

Figures 7.7-a and 7.7-b illustrate the performance of the two controllers to both a step change in the desired concentration of the product R (from 35.0#/ft$^3$ to 40.0#/ft$^3$) and to a step change in load ($CA_0 = 0.0#$/ft$^3$ to $CA_0 = 1.0#$/ft$^3$). (In both cases the complete predictor algorithm including both the measured and calculated disturbance terms ad indicated by Equation (7.16) was used.) Note in both cases the predictor algorithm performs much better than the conven-
Figure 7.7. Computer Control of a Non-Ideal Plug Flow Reactor.
Figure 7.8-a. Set-Point Change

Figure 7.8-b. Load Change

Figure 7.8-b. Control of a Non-Ideal Plug Flow Reactor Using P-A and PI Algorithms.
tional controller. This success of the predictor algorithm occurs in spite of the fact that the actual process is much more complex than the assumed first order lag plus dead time. The complexity is indicated by the fact that the initial response to an increase in manipulated variable (concentration of A) is a decrease in the reaction rate (due to the reduction in residence time and cooling effect of the corresponding increase in flow rate).

Summary

This chapter demonstrates some of the potential which exists in direct digital control by the use of algorithms designed specifically for implementation on a digital computer instead of using the conventional one-, two-, and three-mode controllers.

The predictor algorithm developed and illustrated in this chapter make use of the computer memory capabilities and computing power of a digital computer to effectively eliminate the detrimental effect of the process dead time and the sampled data interface for the control of a process described by a first order lag plus dead time. The results are a digital algorithm which has the following advantages over conventional control:

1. A single algorithm performs both dynamic feedforward and feedback control.
2. The response is not effected by the process dead time.
3. The response behaves much the same regardless the sampling time.
4. The proportional and reset modes are "separated" resulting in integral action only when an unmeasured disturbance enters the system.
5. The performance does not deteriorate with the presence of physical constraints on the allowable control action.

6. The proportional-predictor and the integral-predictor modes can be "tuned" analytically from a knowledge of the process parameters.

7. The performance is quite good even with the presence of severe modeling errors.
NOMENCLATURE

\begin{align*}
G & = \text{Process gain} \\
\tau_1, \tau_2 & = \text{Process time constants} \\
\theta & = \text{Process dead time} \\
T & = \text{Sampling time} \\
N & = \text{Integer number of sampling times in } \theta \\
\beta & = \text{Portion of dead time } \theta \text{ which cannot be expressed in terms of } N \\
SP & = \text{Set-point} \\
\hat{R}_0 & = \text{Calibrated set-point} \\
K_R & = \text{Set-point calibration} \\
K_C & = \text{Controller proportional gain} \\
K_I & = \text{Controller integral gain} \\
\bar{x}_i & = \text{Process output sampled at time } iT \\
\bar{u}_i & = \text{Manipulated variable calculated at time } iT \\
\bar{Z}_i & = \text{Measurable disturbance entering the system at time } iT \\
\bar{d}_0 & = \text{Disturbance calculated from the integral predictor at time zero}
\end{align*}
LITERATURE CITED


CHAPTER VIII

CONCLUSION

In this work the emphasis has been upon the practical application of the digital computer to the control of chemical processes. In Chapter II a graphical method is presented from which to determine the economic effect of particular hardware element on the system performance. Hopefully this factor can be considered along with capital cost in the selection and/or design of control loop hardware.

Chapter III focuses on another aspect of the digital control loop. It considers the digital interface and demonstrates that under normal operations the interface can be approximate by a pure dead time of one half the sampling time. This approximation is significant in that it enables continuous design procedures such as hardware sizing and controller tuning to be applied to the sampled data control loop.

Chapters IV and VII demonstrate two algorithms designed for chemical process applications. Chapter IV shows that the high order under-damped system with constraints can be controlled successfully by approximating the system by a second order lag plus dead time and applying the Smith-predictor and a second order time-optimal switching curve.

Chapter VII on the other hand presents a general digital control algorithm design specifically for the control of a first order lag plus dead time. The techniques demonstrates that a rather sophisticated combination of feedforward and feedback control can be designed.
and represented by a relatively simple digital algorithm. This algorithm results in a significant improvement over the conventional one-, two-, and three-mode algorithm, which in the past have been the mainstay for both analog and digital feedback control.

Recently much emphasis has been placed on model regression both, on-line and off-line. Chapter V and VI both consider this problem and illustrate two different methods by which the non-linear least square regression can be formulated for on-line analysis. Chapter V illustrates how specifying the system to be forced by a perfect pulse reduces the regression program requirements both in terms of running time and computer storage. Chapter VI demonstrates how for most models the multidimensional search can be eliminated and how the use of the "weight average" estimate of the product and cross-product terms will result in a regression which can be applied continuously to closed loop data. Both methods could be used with the algorithms described in Chapter IV and VII to update the system for changes in process parameters.
This appendix contains a description of the implementation and use of subroutine PATERN. This subroutine was written to support the multidimensional optimization required in the previous studies. The program is available through IBM Share Library, titled Multidimensional Optimization Using Pattern Search, PID no. 6248, 1968.
Subroutine PATERN

Purpose

PATTERN is a complete numerical search technique employing essentially a Pattern Search strategy (Wilde, Optimum Seeking Methods, Prentice-Hall, 1964). It can be used to minimize a unimodal function of as many as 1000 independent variables and subject to an unlimited number of nonlinear constraints.

Description

PATTERN is a deceptively simple program for use in a numerical search. Unlike other more sophisticated techniques it does not evaluate or attempt to evaluate the gradient of the function being searched, nor does it ever perform a one dimensional search along the gradient or any coordinate direction as do most other techniques. It simply moves from point-to-point in a somewhat crude fashion asking only one question after each function evaluation. "Does this point yield an improvement"? However, much to the dismay of the exponents of the mathematically orientated techniques the pattern search strategy consistently provides a reliable, fast, efficient means to search complex functions. The strength of PATTERN seems to be entirely in its simplicity. The very fact it does not ever require a gradient surprisingly seems to offer considerable advantage in most complex functions and especially for a search involving more than two variables. The principle handicap placed on the gradient techniques is the shear number of function evaluations required to numerically evaluate the gradient at a point. PATTERN, on the otherhand, does not tarry at
any one point; while the gradient techniques are busy evaluating the
gradient, PATERN is free to move toward the optimum. Granted, if the
value of the gradient provided a real clue to the location of the op­
timum the effort spent evaluating the gradient would be well worth­
while. However, if the function contains valleys and ridges the value
of the direction of steepest descent (the direction of the gradient)
becomes questionable. If the valleys and ridges are very steep the
direction predicted by the gradient might be completely erroneous.
PATERN, in general, is insensitive to the severity of the contours.
Steep valleys which normally give gradient techniques fits are gener­
ally handled with relative ease by pattern logic.

The simplicity of PATERN also enables the handling of hard con­
straints on the search parameters with relative ease. Gradient tech­
niques which have hard constraints must invariably determine a gradient
on the boundary. This causes rather severe numerical problems which
must somehow be circumvented. Typically, the additional logic required
to accomplish this becomes quite involved and sometimes even surpasses
the original unconstrained problem in complexity.

Implementation of a Digital Search Using PATERN

In order to use PATERN in a digital search, three programs are
required besides subroutine PATERN as shown in the Macro flow chart on
the following page. A main program is required to define all the nec­
essary search parameters and to call PATERN. Subroutine PATERN then
conducts a complete search returning to the main program only after
the optimum has been found. The function to be searched by PATERN is
contained in the subroutine PROC. This subroutine must be written
MACRO FLOW CHART - PATTERN SEARCH

MAIN PROGRAM
Contains the overall logic peculiar to the particular study being made. It also defines all the initial search parameters and calls subroutine PATTERN.

PATTERN
Optimization Subroutine which conducts the search specified in the main program and RETURNS only after the optimum answer has been determined.

PROC
Subroutine called by PATTERN. It contains the particular function to be searched. Given a value for a set of search parameters PROC calculates a corresponding value of cost.

BOUNDS
Subroutine called by PATTERN to determine if a particular set of search parameters has violated any constraint. It contains all the constraint equations and only answers the question "yes" or "no".

Note:
Subroutine PATTERN contains all the necessary logic required to perform a particular optimization. It also provides several output options which may be used to print the final result or may be used to follow the search in detail. It is essentially self-contained and is in ready-to-use form. However, PROC and BOUNDS are both subroutines which must be written for the particular problem under study.

Figure A.1. Macro Flow Chart - Multidimensional Optimization using Subroutine PATTERN.
specifically for each individual study. The only convention which must be met in writing the program is that it will, given a particular set of independent variables, calculate the corresponding cost (or value of the search criterion). Subroutine BOUNDS is also a subroutine which must be written specifically for each individual study. It should contain all the equations and should then tell PATERN simply if the boundary has been violated.

**Call Statement Necessary Parameters**

The following is a general call statement used to initiate PATERN

CALL PATERN (NP, P, STEP, N PASS, IO, COST)

**NP** - Number of parameters to be searched

The maximum possible is 1000.

**P** - A vector with NP elements which contains independent variables which are varied during the search.

**STEP** - A vector with NP elements which contain the initial step size corresponding to each parameter.

**NPASS** - Each time PATERN finds an optimum for a particular step size the program can either accept the answer as the optimum or reduce the step size for each parameter and re-initiate the search. NRD corresponds to the number of passes through PATERN before the answer is accepted as optimum.

**IO** - Specifies the print out option available in PATERN

0 - No output

1 - Only the Final answer

2 - The results of each iteration are printed

3 - The results of each process evaluation are printed
**COST** - The cost corresponding to the minimum cost found by PATERN

Note: The parameters NP, P, STEP, NPASS and IO all must be defined before PATERN is called. The initial value of the vector P should be the best guess of the optimum and will be the point around which the search will begin. The values given STEP and NPASS will vary from problem to problem. For a problem in which little is known about where the optimum will lie STEP should be large and NPASS large. If the starting point is known to be close to the optimum it would be better to start with small values of both NPASS and the vector STEP. The relative values of the elements of STEP will depend on some prior knowledge of the relative effect of each parameter on the function to be minimized. If no prior knowledge exists all elements should be made equal.

**Writing PROC and BOUNDS**

The only requirement in writing Subroutine PROC is that given the vector, P, it calculates the corresponding cost. The call statement used by PATERN is the following:

```
CALL PROC (P, COST)
```

*P* is the vector containing the search parameters

*COST* is the value of the function to be minimized where

```
COST = f(P)
```

Subroutine BOUNDS must be written to contain all the constraints. It simply checks each constraint equation and tells PATERN "yes a constraint has been violated" or says "no violation". The call statement used by PATERN is the following:
CALL BOUNDS (P, IOUT)

If IOUT = 1 a violation has occurred
If IOUT = 0 no violation has occurred

Even in studies in which the search is unbounded, subroutine BOUNDS is still necessary and should be as follows:

SUBROUTINE BOUNDS (P, IOUT)

DIMENSION P ( )

IOUT = 0

RETURN

END.

Example

The following is a simple example to help illustrate how to write the necessary programs.

Consider the following three dimensional search. The problem is to minimize the following function

$$\text{COST} = x_1^2 + x_2^2 + x_3^2$$

subject to the following constraints

$$x_1 \geq 0$$

$$x_2 \leq 10$$

$$x_1 + x_3 \geq x_2$$
**COST** - The cost corresponding to the minimum cost found by PATERN

Note: The parameters NP, P, STEP, NPASS and IO all must be defined before PATERN is called. The initial value of the vector P should be the best guess of the optimum and will be the point around which the search will begin. The values given STEP and NPASS will vary from problem to problem. For a problem in which little is known about where the optimum will lie STEP should be large and NPASS large. If the starting point is known to be close to the optimum it would be better to start with small values of both NPASS and the vector STEP. The relative values of the elements of STEP will depend on some prior knowledge of the relative effect of each parameter on the function to be minimized. If no prior knowledge exists all elements should be made equal.

**Writing PROC and BOUNDS**

The only requirement in writing Subroutine PROC is that given the vector, P, it calculates the corresponding cost. The call statement used by PATERN is the following

```
CALL PROC (P, COST)
```

P is the vector containing the search parameters

COST is the value of the function to be minimized where

```
COST = f(P)
```

Subroutine BOUNDS must be written to contain all the constraints. It simply checks each constraint equation and tells PATERN "yes a constraint has been violated" or says "no violation". The call statement used by PATERN is the following
Main Program:

DIMENSION P (3), STEP (3)
P(1) = 5.0
P(2) = 5.0  Starting Point
P(3) = 5.0

STEP (1) = 1.0
STEP (2) = 1.0  Initial step size
STEP (3) = 1.0

CALL PATTERN (3, P, STEP, 3, 2, COST)
STOP
END

SUBROUTINE PROC (P, COST)
DIMENSION P(3)
COST = P(1)**2 + P(2)**2 + P(3)**2
RETURN
END

SUBROUTINE BOUNDS (P, IOUT)
DIMENSION P(3)
IOUT = 0
IF(P(1).LT. 0.0) IOUT = 1
IF(P(1).GT. 10.) IOUT = 1  Constraints
IF((P(1) + P(3)).LT. P(2)) IOUT = 1
RETURN
END
SUBROUTINE PATTERN(NP,P,STEP,NPASS,IO,COST)

PATTERN SEARCH

*GENERAL MULTIVARIABLE SEARCH PROGRAM TO MINIMIZE A COST FUNCTION
*USING PATTERN SEARCH MODIFIED TO INCLUDE CONSTRAINTS.

*DEFINITIONS

NP----NUMBER OF PARAMETERS TO BE SEARCHED (INTEGER)
P----PARAMETERS TO BE SEARCHED (VECTOR OF LENGTH NP)
STEP—INITIAL STEP SIZE OF EACH PARAMETER (VECTOR OF LENGTH NP)
NPASS—NUMBER OF PASSES THROUGH PATTERN WITH THE STEP SIZE OF EACH
PARAMETER REDUCED BY A FACTOR OF 10
IO----OUTPUT OPTION AVAILABLE IN SUBROUTINE PATTERN
IO=0....NO OUTPUT
IO=1....FINAL ANSWER ONLY, PRINTED
IO=2....RESULTS OF EACH ITERATION PRINTED
IO=3....RESULTS OF EACH STEP PRINTED
COST—CURRENT VALUE OF THE CRITERION FUNCTION BEING MINIMIZED

*SUPPORTING PROGRAMS WHICH MUST BE WRITTEN BY USER

BOUNDS(P,IOUT)---SUBROUTINE WRITTEN BY USER TO CHECK FOR BOUNDARY
VIOLATIONS FOR A PARTICULAR VALUE OF THE VECTOR P. PROGRAM
SHOULD BE WRITTEN SO WHEN A VIOLATION OCCURS IOJT IS SET EQUAL TO ONE AND WHEN NO VIOLATION OCCURS SET EQUAL TO ZERO
PROC(P,COST)---SUBROUTINE WRITTEN BY USER TO CALCULATE A VALUE OF COST FOR A PARTICULAR SET OF PARAMETERS, P.

DIMENSION P(NP),STEP(NP),B1(1000),B2(1000),T(1000),S(1000)

NOTE***VECTORS B1,B2,T,S NEED ONLY BE DIMENSIONED BY A NUMBER EQUAL TO THE NUMBER OF PARAMETERS, NP.

STARTING POINT
NRD=NPass
L=1
ICK=2
ITTER=0
DO5 I=1,NP
B1(I)=P(I)
B2(I)=P(I)
T(I)=P(I)
5 S(I)=STEP(I)*10.

INITIAL BOUNDARY CHECK AND COST EVALUATION
CALL BOUNDS(P,IOUT)
IF(IOUT.LE.0)GOTO7
6 RETURN
10 CALL PROC(P,C1)
   IF(I0.LE.1)GOTO11
   WRITE(16,1000)J,P(J),J=1,NP
7 RETURN
11 DO99 INRD=1,NRD
DO12  I=1,NP
12  S(I)=S(I)/10.
    IF(IO.LE.1)GOTO20
    WRITE(6,1003)
    WRITE(6,1000)(J,S(J),J=1,NP)
20  IFAIL=0.0
C-----PERTURBATION ABOUT T
DO30  I=1,NP
         IC=0
21  P(I)=T(I)+S(I)
    IC=IC+1
    CALL BOUNDS(P,IOUT)
    IF(IOUT.GT.0)GOTO23
    CALL PROC(P,C2)
         L=L+1
    IF(IO.LT.3)GOTO22
    WRITE(6,1002)L-C2
    WRITE(6,1000)(J,P(J),J=1,NP)
22  IF(C1-C2)23,23,25
23  IF(IC.GE.2)GOTO24
    S(I)=-S(I)
    GOTO21
24  IFAIL=IFAIL+1
    P(I)=T(I)
    GOTO30
25  T(I)=P(I)
    C1=C2
30  CONTINUE
    IF(IFAIL.LT.NP)GOTO35
    IF(ICK.EQ.2)GOTO90
    IF(ICK.EQ.1)GOTO35
    CALL PROC(T,C2)
         L=L+1
    IF(IO.LT.3)GOTO31
WRITE(6,1002)L,C2
WRITE(6,1000)(J,T(J),J=1,NP)

31 IF(C1-C2)32,34,34
32 ICK=1
DO33 I=1,NP
B1(I)=B2(I)
P(I)=B2(I)
33 T(I)=B2(I)
GOTO20
34 C1=C2
35 IB1=0
DO39 I=1,NP
B2(I)=T(I)
IF(ABS(B1(I)-B2(I)).LT.0.01*ABS(S(I))) IB1=IB1+1
39 CONTINUE
IF(IB1.EQ.NP)GOTO90
ICK=0
ITER=ITER+1
IF(IO.LT.2)GOTO40
WRITE(6,1001)ITER,C1
WRITE(6,1000)(J,T(J),J=1,NP)

C-----ACCELERATION STEP
40 SJ=1.0
DO45 II=1,11
DO42 I=1,NP
42 P(I)=T(I)
SJ=SJ-.1
CALL BOUNDS(T,IOUT)
IF(IOUT.LT.1)GOTO46
IF(II.EQ.11)ICK=1
45 CONTINUE
46 DO47 I=1,NP
47 B1(I)=B2(I)
GO TO 20

90  DO 91 I=1,NP
91   T(I)=82(I)
99   CONTINUE
        DO 100 I=1,NP
100  P(I)=T(I)
        COST=C1

          IF(I0.LE.0)RETURN
          WRITE(6,1004)L,C1
          WRITE(6,1000)(J,P(J),J=1,NP)

          RETURN

1000  FORMAT(10X,5(I7,E13.6)/)
1001  FORMAT(1X,14HITTERATION NO. ,15/5X,5HCOST= ,E15.6,20X,
1             10HPARAMETERS)
1002  FORMAT(10X3HNO. ,14, 8X5HCOST=,E15.6)
1003  FORMAT(1X28HSTEP SIZE FOR EACH PARAMETER )
1004  FORMAT(1H113HANSWERS AFTER ,13,2X,23HFUNCTIONAL EVALUATIONS //
1             5X5HCOST=,E15.6,20X,18HOPTIMAL PARAMETERS )
1005  FORMAT(1H135HINITIAL PARAMETERS OUT OF BOUNDS )
END
APPENDIX B

Figures - Chapter II
Figure B-1. Effect of Dimensionless Hardware Lag on System Response for a Continuous System Tuned by ISE.
Figure B-2. Effect of Dimensionless Hardware Lag on System Response for a Continuous System Tuned by ITAE.
Figure B-3. Effect of Dimensionless Hardware Lag on System Response for a Continuous System Tuned by IAE.
Figure B-4. Effect of Dimensionless Hardware Lag on System Response for a Sampling Time of 0.10 x $\tau_1$, Tuned by IAE.
Figure B-5. Effect of Dimensionless Hardware Lag on System Performance for a Sampling Time of $0.20 \times \tau_1$, Tuned by IAE.
Figure B-6. Effect of Dimensionless Hardware Lag on System Performance for a Sampling Time of $0.30 \times \tau_1$, Tuned by IAE.
Figure B-7. Effect of Dimensionless Hardware Lag on System Response for a Sampling Time of 0.40 x $\tau_1$, Tuned by IAE.
Figure B-8. Effect of Dimensionless Hardware Lag on System Response for a Sampling Time of 0.50 x $\tau_1$, Tuned by IAE.
Figure B-9. Error Ratio Versus Dimensionless Hardware Lag for a Continuous System Tuned by ISE.
Figure B-10. Error Ratio Versus Dimensionless Hardware Lag for a Continuous System Tuned by ITAE.
Figure B-11. Error Ratio Versus Dimensionless Hardware Lag for a Continuous System Tuned by IAE.
Figure B-12. Error Ratio Versus Dimensionless Hardware Lag for a Sampling Time of $0.10 \times \tau_1$, Tuned by IAE.
Figure B-13. Error Ratio Versus Dimensionless Hardware Lag for a Sampling Time of $0.20 \times \tau_1$, Tuned by IAE.
**Figure B-14.** Error Ratio Versus Dimensionless Hardware Lag for a Sampling Time of $0.30 \times \tau_1$, Tuned by IAE.
Figure B-15. Error Ratio Versus Dimensionless Hardware Lag for a Sampling Time of 0.40 x τ₁, Tuned by IAE.
Figure B-16. Error Ratio Versus Dimensionless Hardware Lag for a Sampling Time of $0.40 \times \tau_1$, Tuned by IAE.
Figure B-17. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Continuous System Tuned by ISE.
Figure B-18. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Continuous System Tuned by ITAE.
Figure B-19. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Continuous System Tuned by IAE.
Figure B-20. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of $0.10 \times \tau_1$, Tuned by IAE.
Figure B-21. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of $0.20 \times \tau_1$, Tuned by IAE.
Figure B-22. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of 0.30 x $\tau_1$, Tuned by IAE.
Figure B-23. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of 0.40 x $\tau_1$, Tuned by IAE.
Figure B-24. Dimensionless Hardware Lag Versus Dimensionless Dead Time at Constant Increase in Error for a Sampling Time of $0.50 \times \tau_1$, Tuned by IAE.
APPENDIX C

COMPUTER PROGRAMS
PROGRAM II-1

C * HARDWARE DYNAMICS STUDY
C
C * THIS PROGRAM CO-ORDINATES THE STUDY OF HARDWARE DYNAMICS IN THE
C * CONTROL LOOP. THE STUDY INVOLVES SEARCHING FOR THE OPTIMUM SYSTEM
C * RESPONSE BASED ON IAE, ISE, AND ITAE FOR VARIOUS COMBINATIONS OF
C * SAMPLING TIME, DEAD TIME, AND HARDWARE TIME CONSTANT.
C
DIMENSION XL(2),U(2),PAR(2),STEP(2)
DIMENSION DUM1(100),DUM2(100,10),DUM3(100)
COMMON DUM1,DUM2,DUM3
COMMON NPARMT,TLV,TLP,TDP,SP,TE,D,IPLOT,IPL0T,HSLOPE,HFIRST
1 ,TL,TD,IFFIT,TMP,ST,HLAST
1 ,ICFUN
CALL FPTRAP(-3)
XL(1)=0
XL(2)=0
U(1)=1000.
U(2)=1000.
IFFIT=0
NPARMT=2
IPL0T=1
NXL=1
TLP=1
HSLOPE=.005
HLAST=.025
HFIRST=.001
SP=0.0
PSTRT1=2.0
PSTRT2=2.52
1 CONTINUE
ISTRT=0
READ2000 ,ICFUN,ISR,ST,TDP,TLV,ERO,PSTRT1,PSTRT2,PAR1,PAR2
2000 FORMAT(11,11,F8.3,T10.3)
NLAG1=TLV*10.+1.5
NDEL1=TDP*10.+2.5
IF(TOP .LE.16) NDEL1=3
IF(TOP .LE.11) NDEL1=2
IF(TOP .LE.06) NDEL1=1
D=1.0
PRINT9999
9999 FORMAT(121X3HTDP,18X3HTLV,16X4HGAIN,15X5HRESET,17X3HPHI,15X5HRA
NLDEL0=1
IF(I=ISTRT.EQ.0) NLDEL0=NDEL1
DO100 IDEL=NLDEL0,12
TDP=IDEL-2
TDP=TDP/10.
IF(IDEL.EQ.1) TDP=.05
IF(IDEL.EQ.3) TDP=.15
IF(IDEL.EQ.2) TDP=.10
PAR(1)=PSTRT1
PAR(2)=PSTRT2
IF(I=ISTRT.EQ.0 )PAR(1)=PAR1
IF(I=ISTRT.EQ.0 )PAR(2)=PAR2
NLAGO=1
IF(I=ISTRT.EQ.0)NLAGO=NLAG1
ISTRT=1
DO100 ILAG=NLAGO,11
TLV=ILAG-1
TLV=TLV/10.
TEE = (1.0 + 8.0*TLV + 8.0*TDP)
TE = 2.0 * TEE
TMP = 3.0 * TEE / 120.
IF PLOT = 0
STEP(1) = .1
STEP(2) = .1
NSTEP = 2
IF (ILAG > 3) STEP(1) = .01
IF (ILAG > 3) STEP(2) = .01
IF (ILAG > 3) NSTEP = 1
CALL PATTERN (NPARMT, XL, U, PAR, NSTEP, STEP, PHI)
C CALL OPTGRD (XL, U, PAR, PHI, NLX)
IF (ISR EQ 2) CALL OPTGRD (XL, U, PAR, PHI, NLX)
IF (ISR EQ 1) CALL PATTERN (NPARMT, XL, U, PAR, NSTEP, STEP, PHI)
IF PLOT = 1
IF (ILAG EQ 1) CALL PROC (XL, PAR, CFUN)
IF (ILAG EQ 6) CALL PROC (XL, PAR, CFUN)
IF (ILAG EQ 11) CALL PROC (XL, PAR, CFUN)
IF PLOT = 0
TE = 6.0 * TEE
CALL PROC (XL, PAR, PHI)
IF (ILAG EQ 1) ERO = PHI
ERATO = PHI / ERO
PRINT 1000, TDP, TLV, PAR(1), PAR(2), PHI, ERATO
1000 FORMAT (1X6E19.8)
PUNCH 1001, TDP, TLV, PAR(1), PAR(2), PHI, ERATO
1001 FORMAT (2F5.2, 10X, , 4E15.6)
IF (IDEL > 0) GO TO 100
PSTRT1 = PAR(1)
PSTRT2 = PAR(2)
100 CONTINUE
GOTO 101
6969 STOP
END
SUBROUTINE PROC(NLX, PAR, CFUN)

* PROCESS LOOP—HARDWARE DYNAMICS STUDY

* THIS SUBROUTINE CONTAINS THE PROCESS LOOP USED IN THE HARDWARE
* DYNAMICS STUDY. IT CONSISTS OF A DISCRETE PI CONTROLLER, A PJRE
* FIRST ORDER LAG PLUS DEAD TIME PROCESS. THE PROGRAM ALSO CALCULATES
* THE PERFORMANCE CRITERIA IAE, ISE, ITAE FOR A SET OF PROCESS PARAMETERS
* AND CONTROLLER CONSTANTS.

* 

DIMENSION PSTORE(2,200)
DIMENSION P(1000), PAR(10)
COMMON DUM1(100), DUM2(100,10), DUM3(100)
COMMON NPARMT, TLV, TLP, TDP, SP, TE, O, IFPLOI, IPOINT, HSLOPE, HFIRST
1, TL, TD, IFIT, TMP, ST, HLAST
1, ICFUN
IFPLO=IFPLOI
CALL SSWTCH(4,IFPLOI)
IF(IFPLOI.EQ.1) IFPLOT=1
TLV1=TLV
TLP1=TLP
TDP1=TDP
GP=PAR(1)
GR=PAR(2)

C

C-----INITIAL CONDITIONS
10 ESQ=0.0
ERA=0.0
ERAT=0.0
R1=0.0
T=0.0
STI=ST
TPlot=0.0
IDUM=0
CIN=0.0
COUT=0.0
VOUT=0.0
PIN=0.0
POUT=0.0
ICOUNT=1
K=1
TMEX=0.0
HPlot=0
NPLOT=0
D015 I=1,200
PSTORE(1,I)=0
15 PSTORE(2,I)=0
15D20 J=1,1000
20 P(J)=0.0

C-------TIME GENERATION - REAL AND DELAYED-------
25 H=HSLOPE*T+HFIRST
IF(H.GT.HLAST)H=HLAST
T=T+H
K=K+1

C
C-------IF USING EQUIVALENT FIRST ORDER LAG PLUS DEAD TIME, IFFIT=1
25 IF(IFFIT.EQ.1) TLP=TL
25 IF(IFFIT.EQ.1) TLV=H
25 IF(IFFIT.EQ.1) TDP=TD

C
C-----CALC. OF DELAY TIME
        IF(TDP.GT.H)GOTO30
        K1=K-1
        DO26 I=1,20
           IJI=I-1
           IF(K1-I*1000.LE.0)GOTO27
26       CONTINUE
27       K1=K1-IJI*1000
        PDLY=P(K1)
        GOTO35
30       TME=T-TDP
        IF(TME)31,31,32
31       PDLY=0.0
        GOTO35
32       ICOUNT=ICOUNT+1
        DTMEX=HSLOPE*TMEX+HFIRST
        TMEXP=TMEX
        IF(DTMEX.GT.HLAST)DTMEX=HLAST
        TMEX=TMEXP+DTMEX
        IF(TMEX.LT.TME)GOTO32
        ICOUN=ICOUNT
        DO33 I=1,20
           IJI=I-1
           IF(ICOUN-I*1000.LE.0)GOTO34
33       CONTINUE
34       ICOUNT=ICOUNT-IJI*1000
        ICOUN1=ICOUNT-1
        IF(ICOUN.EQ.1)ICOUN1=1000
        PDLY=(P(ICOUN )-P(ICOUN1))*(TME-TMEXP)/DTMEX+P(ICOUN1)
C
C------COMPARATOR - ACCUMULATED ERROR
35       CIN=SP-PDLY
        E=CIN*H
        ERA=ERA+ABS(E)
ERAT=ERAT+ABS(E)*T
ESQ=ESQ+H*CIN**2
IF(ESQ.GE.1.0E10)GOTO66
C
C------SAMPLE AND HOLD
IF(ST.LE.H)ST=H
STI=STI+H
IF(ST-STI)41,41,45
41 STI=0.0
C
C------CONTROLLER
42 R1=R1+CIN*ST*GR*GP
43 COUT=GP*CIN+R1
VIN=COUT
C
C------VALVE
45 IF(TLV-H)46,46,47
46 VOUT=VIN
GOTO50
47 VOUT=VOUT+(VIN-VOUT)*H/TLV
C
C------DISTURBANCE
50 PIN=VOUT+D
C
C------PROCESS
55 IF(TLP-H)56,56,57
56 POUT=PIN
GOTO60
57 POUT=POUT+(PIN-POUT)*H/TLP
60 KK=K
DO58 I=1,20
IJI=I-1
IF(KK-I*1000.LE.0)GOTO59
58 CONTINUE
59  KK=K-IJI=1000
   P(KK)=POUT
C
C------PROCESS DEAD TIME CALC. IN COMPARATOR
C
C------STORING VALUE TO BE PLOTTED
IF(IFPLOT.EQ.0)GOTO65
   TPL0T=TPL0T+H
IF(TPL0T-TMP.LT.0.0)GOTO65
   TPL0T=0
   NPL0T=NPL0T+1
   PSTORE(1,NPL0T)=T
IF(IPL0T.EQ.1)PSTORE(2,NPL0T)=POUT
IF(IPL0T.EQ.2)PSTORE(2,NPL0T)=COUT
IF(IPL0T.EQ.3)PSTORE(2,NPL0T)=VOUT
65  IF(T-TE)25,70,70
66  CFUN=1.0+ABS(TE-T)*1.0E10)
   GOTO71
70  IF(ICFUN.EQ.1)CFUN=ERA
IF(ICFUN.EQ.2)CFUN=ERAT
IF(ICFUN.EQ.3)CFUN=ESQ
IF(IFPLOT.EQ.1)CALL PLOT1(PSTORE,NPL0T,GP,GR,ESQ)
71  IF(ST-H.LE.-02)ST=0.0
RETURN
END
SUBROUTINE PRPLOT(T,P,PO,PM,TITLE,NAME,VALUE,N,IDUM)

DATA BLANK,STAR,DASH,DOT,ZERO,ONE,X,U
  X /1H ,1H*,1H-,1H.,1H0,1H1,1H+,1HU/
  DIMENSION CHAR(120),NAME(N),VALUE(N),TITLE(5)

T = TIME (X AXIS)
N = NUMBER OF PARAMETERS TO BE IDENTIFIED
IDUM = DUMMY VARIABLE TO INITIATE ROUTINE
P = FUNCTIONAL VALUE TO BE PLOTTED (FULL SCALE=PM)
NAME = 5 LETTER NAMES OF PARAMETERS TO BE IDENTIFIED
VALUE = VALUE OF PARAMETER TO BE IDENTIFIED

DRAWING TIME=0 AXIS
IF(IDUM.GT.1)GOTO40
10 CHAR(1)=ZERO
   PRINT 5,TITLE, PM
   FORMAT(1H1,35X5A6,10X6H(MAX=,E10.3,1H))
   MT=0
   DO33 K2=2,120
33 CHAR(K2)=DASH
   IFORMT=1
GOTO90
C------DRAWING RESPONSE AXIS
40   DO41 K3=1,120
41   CHAR(K3)=BLANK
     MT=MT+1
45   DO45 K4=10,120,10
45   CHAR(K4)=DOT
C------DRAWING RESPONSE=0 AXIS
KO=PO+.1
CHAR(KO)=ZERO
C------PLOTTING POINTS(POINTS OFF-SCALE LABELED U)
50   IF(ABS(P-3333.)>.1)60,60,53
53   SCALE=110.-PO
     I=SCALE*P/PM+PO+.5
     IF(I.LT.2)GOTD55
     IF(I.GT.120)GOTD56
     CHAR(I)=X
     GOTO60
55   CHAR(2)=U
     GOTO60
56   CHAR(120)=U
C------LABLING KEY AT 1 TIME UNIT
60   IF(IDUM-2)61,61,65
61   IN=-1
65   IF(IN-N)69,70,66
66   IFORMT=1
     GOTO80
69   IN=IN+1
     IF(IN.EQ.0)GOTD71
     IFORMT=3
     GOTO80
70   IN=IN+1
71   DO75 K5=100,120
75   CHAR(K5)=STAR
IFORMT=1

C------LABLING TIME AXIS
80 IF(MT-10)90,85,90
85 CHAR(J9)=DASH
   IFORMT=IFORMT+1
   MT=0

C------OUTPUT
90 IF(IFORMT.EQ.1)PRINT101,CHAR
   IF(IFORMT.EQ.2)PRINT 102,T,(CHAR(J2),J2=9,120)
   IF(IFORMT.EQ.3)PRINT 103,(CHAR(J3),J3=1,99),NAME(IN),VALUE(IN)
   IF(IFORMT.EQ.4)PRINT104,T,(CHAR(J4),J4=9,99),NAME(IN),VALUE(IN)
101 FORMAT(120A1)
102 FORMAT(1X,F7.3,112A1)
103 FORMAT(99A1,2H*,A6,1H=E10.3,2H *)
104 FORMAT((1X,F7.3,91A1,2H*,A6,1H=E10.3,2H *)
RETURN
END

SUBROUTINE PLOT1(PSTORE,NPLOT,GP,GR,ESQ)
DIMENSION PSTORE(2,200)
DIMENSION NAME(7),VALUE(7),TITLE(5),TITLE1(5),TITLE2(5),TITLE3(5)
COMMON DUM1(100),DUM2(100,10),DUM3(100)
COMMON NPARMT,TLV,TLP,TDP,SP,TE,D,IFPLOT,IPL,TITLE1,TITLE2,TITLE3
   TL,TD,IFFIT,TMP,ST,HLAST
DATA NAME/42HLAGV LAGP DELAY PROP RESET ERROR STIME /
DATA TITLE1/30H PROCESS RESPONSE /
DATA TITLE2/30H CONTROLLER RESPONSE /
DATA TITLE3/30H VALVE RESPONSE /
VALUE(1)=TLV
   VALUE(2)=TLP
   VALUE(3)=TDP
   VALUE(4)=GP
   VALUE(5)=GR
   VALUE(6)=ESQ
   VALUE(7)=ST
PM=0
DO10 I=1,200
10 IF (ABS (PSTORE (2, I)) - ABS (PM) .GT. 0.) PM = PSTORE (2, I)
N=7
PO=50
DO20 I=1,5
   IF (IPL0T.EQ.1) TITLE(I)=TITLE1(I)
   IF (IPL0T.EQ.2) TITLE(I)=TITLE2(I)
   IF (IPL0T.EQ.3) TITLE(I)=TITLE3(I)
20 CONTINUE
DO30 I=1,NPLOT
   T=PSTORE (1, I)
   PLT=PSTORE (2, I)
   CALL PRPLOT (T, PLT, PO, PM, TITLE, NAME, VALUE, N, I)
30 CONTINUE
CONTINUE
RETURN
END
PROGRAM II-4

SUBROUTINE OPTGRD(XL, U, P, PHI, NLX)

* OPTIMUM GRADIENT SEARCH

* THIS SUBROUTINE CONTAINS THE LOGIC FOR OPTIMUM GRADIENT SEARCH. IT
* IS USED TO SPOT CHECK THE RESULTS OF THE OPTIMIZATION USING PATTERN
* SEARCH

DIMENSION P(10),PL(10),P2(10),PM(10),DP(10),DPM(10),PO(10),GPHI(10
1),U(10),XL(10)
NPARMT=2
CALL SSWTCH(1,ID)

OPTIMIZATION BY GRADIENT SEARCH
P VECTOR IS PARAMETER VECTOR
U VECTOR CONTAINS UPPER BOUND ON PARAMETER VECTOR
XL VECTOR CONTAINS LOWER BOUNDS ON PARAMETER VECTOR
GPHI=GRADIENT VECTOR OF CRITERION FUNCTION
P(1)=CONTROLLER GAIN
P(2)=RESET TIME
DP VECTOR CONTAINS CHANGES IN P VECTOR
W VECTOR IS WEIGHTING FACTOR FOR CRITERION FUNCTION
ASSUME ALL INTERVALS ARE WEIGHTED EQUALLY, WITH WEIGHT 1
NOTE—IF UPPER AND LOWER BOUND ON PARAMETER ARE EQUAL THEN
PARAMETER IS KEPT CONSTANT.
NOTE—DO NOT INITIALIZE PARAMETERS OUTSIDE BOUNDS
EPS=.0001
C READ IN CONSTANTS
C
STO=1.
C
C DESIRED RESPONSE DATA GENERATION
C AND GENERATION WEIGHT FUNCTION
C ITAE CRITERION FOR A STEP INPUT
C
NI=-1
N=0
10 NI=NI+1
20 CALL PROC(NLX,P,PHI)
NLX=2
IF(NI.EQ.0) GOTO1111
IF(ABS(STO-PHI)/STU.LT.TEPS)GO TO 1
1111 STO=PHI
C
C CALCULATION OF GRADIENT ANO MAGNITUDE OF GRADIENT SQUARED
C
DO 21 I=1,NPARMT
21 P2(I)=P(I)
XMAG2=0.
DO 22 I=1,NPARMT
DELTA=ABS(.0001*P(I))
IF(Delta.LT..000001) DELTA=.000001
P2(I)=P(I)+DELTA
IF(P2(I).GT.U(I))GO TO 22
IF(P2(I).LT.XL(I))GO TO 22
CALL PROC(NLX,P2,PHI)
GPHI(I)=(PHI1-PHI)/DELTA
XMAG2=XMAG2+GPHI(I)**2
22 P2(I)=P(I)
GOTO(23,24),10
23 WRITE(6,997) NI,PHI,(I,P(I),GPHI(I),I=1,NPARMT)
24 CALL SSWTCH(2,ITSS)
     GO TO(25,26),ITSS
25 STOP

**CALCULATION OF OPTIMUM STEP SIZE**

26 DO 30 I=1,NPARMT
     DP(I)=-PHI*GPHI(I)/(2.*N*XMG2)
30 P2(I)=P(I)+DP(I)
     CALL PARBOD(U, XL, P2)
     CALL PROC(NLX, P2, PHI2)
40 DO 50 I=1,NPARMT
     DP(I)=.5*DP(I)
50 P1(I)=P(I)+DP(I)
     CALL PARBOD(U, XL, P1)
     CALL PROC(NLX, P1, PHI1)
     IF(PHI1-PHI2.LE.0.)GO TO 171
     IF(PHI2-PHI1.LT.0.)GO TO 60
     PHI2=PHI1
     N=N+1
     GO TO 40
60 IF(N.LE.0.)GO TO 250
     DO 70 I=1,NPARMT
70 DP(I)=4.*DP(I)
80 DO 90 I=1,NPARMT
90 P2(I)=P(I)+DP(I)
     CALL PARBOD(U, XL, P2)
     PHI0=PHI1
     PHI1=PHI2
     CALL PROC(NLX, P2, PHI2)
     N=N-1
     IF(PHI1-PHI2.LT.0.)GO TO 160
     IF(N.LE.0.)GO TO 130
     DO 120 I=1,NPARMT
120  DP(I)=2.*DP(I)
GO TO 80
130  PHI=PHI2
140  DO 150  I=1,NPARMT
150  P(I)=P(I)+DP(I)
CALL PARBOD(U,XL,P)
GO TO 10
160  N=N+1
DO 170  I=1,NPARMT
170  DP(I)=.5*DP(I)
GO TO 210
171  N=N+1
DO 172  I=1,NPARMT
DP(I)=.5*DP(I)
172  PO(I)=P(I)+DP(I)
CALL PARBOD(U,XL,PO)
CALL PROC(NLX,PO,PHIO)
IF(PHI1-PHI0.LE.0.)GO TO 190
180  PHI2=PHI1
PHI1=PHI0
GO TO 171
190  IF(PHI1-PHI.GT.0.)GO TO 180
DO 200  I=1,NPARMT
200  DP(I)=2.*DP11)
210  DO 220  I=1,NPARMT
DPM(I)=.75*DP(I)*(PHI2-5.*PHI1+4.*PHIO)/(PHI2-3.*PHI1+2.*PHIO)
220  PM(I)=P(I)+DPM(I)
CALL PARBOD(U,XL,PM)
CALL PROC(NLX,PM,PHIM)
IF(PHIM-PHI1.GE.0.)GO TO 240
DO 230  I=1,NPARMT
230  PI 1=P(I)+DPM(I)
CALL PARBOD(U,XL,P)
PHI=PHIM
GO TO 10
240 PHI=PHI1
GO TO 140
250 PHI=PHI2
DO 260 I=1,NPARMT
260 P(I)=P(I)+2.*DP(I)
   CALL PARBOD(U,XL,P)
GO TO 10
999 FORMAT(3/(3F10.0))
997 FORMAT(15HOITERATION NO. 13,5X19HCRI TERION FUNCTION E26.5/
   1 10H COMPONENT5X9HPARAMETER5X8HGRADIENT/(1X1S,E18.5,E15.5))
996 FORMAT((//)
1 995 FORMAT(5H1$JOB)
994 FORMAT(/)
993 FORMAT(54X,20HBOUNDS ON PARAMETERS//46X,E11.5,1X,14HG.E. K C L.
1E.,1X,E11.5/46X,E11.5,1X,14HG.E. TR L.E.,1X,E11.5 )
1 RETURN
END
SUBROUTINE PARBOD(U,XL,PT)
   NPARMT=2
C THIS SUBROUTINE KEEPS PARAMETERS WITHIN THEIR SPECIFIED BOUNDS.
DIMENSION U(10),XL(10),PT(10)
DO 10 I=1,NPARMT
   IF(PT(I).GT.U(I))PT(I)=U(I)
   IF(PT(I).LT.XL(I))PT(I)=XL(I)
10 CONTINUE
RETURN
END
PROGRAM III-1

OPEN LOOP RESPONSE - DEAD TIME APPROXIMATION

* THIS PROGRAM COMPARES, AT VARIOUS SAMPLING TIMES, THE OPEN LOOP RESPONSE OF THE SAMPLER AND ZERO ORDER HOLD WITH THE RESPONSE OF A PURE DEAD TIME OF ONE-HALF THE SAMPLING TIME. IN EACH CASE THE OUTPUT IS FILTERED BY A FIRST ORDER LAG.

DIMENSION XI(1002), X(1002), XS(1002), YC(1002), YS(1002), TIME(1002), BUFFER(5000)

CALL PLOTS(BUFFER, 5000)
CALL FPTRAP(1-3)
CALL FACTOR(.90)

READ 1000, NSTOP, TAU1, TAU2, TS
1000 FORMAT(I1, F9.0, 2F10.0)

H=.01
NHOLD=TS/H+.5
NDEL=TS/(2.0*H)+.5

C----- INITIAL CONDITIONS
T=0
I=0
X0=0
X1=0
XH=0.0
YC0=0
YS0=0
DO10 II=1, NDEL
10  XC(I1)=0.0
    IHOLD=0
C-----INTEGRATION LOOP
15  I=I+1
    T=T+H
    TIME(I)=T
    X0=X0+H/TAU1*(1.0-X0)
    X1=X1+H/TAU1*(X0-X1)
    X(I)=X1
    IND=I+NDEL
    XC(IND)=X1
    IF(IHOLD.LT.NHOLD)G0T020
    IHOLD=0
    XH=X1
20  XS(I)=XH
    IHOLD=IHOLD+1
C-----CONTINUOUS APPROXIMATION
    YCO=YCO+H/TAU2*(XC(I)-YCO)
    YC(I)=YCO
C-----SAMPLED DATA
    YSO=YSO+H/TAU2*(XS(I)-YSO)
    YS(I)=YSO
6900  FORMAT(1X,E20.6,2(5X,2E20.6))
    IF(I.LT.1002)G0T015
C-----PLOTTING PROGRAM
11=1001
12=1002
X(I1)=0.0
XC(I1)=0.0
YC(I1)=0.0
XS(I1)=0.0
YS(I1)=0.0
TIME(I1)=0.0
X(I2)=.25
XC(I2)=.25
XSI(I2)*.25
YC(I2)=.25
YSI(I2)*.25
TIME(I2)=2.0
CALL AXIS(0,0,0,0,8HRESPONSE,8,4,0,90,0,0,.25,10.)
CALL AXIS(0,0,0,4HTIME,-4,5,0,0,0,0,0,0,2,0,10.)
CALL LINE(TIME,YS,1000,1,0,11)
CALL LINE(TIME,YC,1000,1,0,11)
CALL LINE(TIME,XS,1000,1,0,11)
CALL LINE(TIME,XC,1000,1,0,11)
CALL PLOT(11,0,0,-3)
IF(NSTOP.EQ.0)GOTO1
CALL PLOT(0,0,0,999)
STOP
END
PROGRAM III-2

DIMENSION P(2), STEP(2)
DIMENSION BUFFER(5000)
CALL PLOTS(BUFFER, 5000)
H = .01
TAU1 = 1.0
CALL FACTOR(.80)
1 READ1000, TAU2, DEL, TS, P(1), P(2)
PRINT1001, TAU2, DEL, TS, P(1), P(2)
1000 FORMAT(5F10.0)
1001 FORMAT(1H1, 5F20.6)
1010 DO100 IR = 1, 2
10 IF(IR.EQ.1) IRUN = 2
11 IF(IR.EQ.2) IRUN = 1
12 IMAX = 2000
13 GOTO(10, 20), IRUN
10 STEP(1) = 1.0
20 STEP(2) = 1.0
NRD=3
GOTO30
20   STEP(1)=.1
     STEP(2)=.1
     NRD=2
30   CALL PATTERN(2,P,STEP,NRD,2,CFUN)
     CALL PROC(P,CFUN)
     IMAX=1200
     CALL AXIS(0.0,0.0,9HRESPONSE,8,4.0,0.0,0.0,50,10.)
     CALL AXIS(0.0,0.0,4HTIME,-4,6.0,0.0,0.0,2.0,10.)
     X(IMAX+1)=0.0
     X(IMAX+2)=.5
     TIME(IMAX+1)=0.0
     TIME(IMAX+2)=2.0
     CALL LINEC TIME,X,IMAX,1,0,1)
     IF(IRUN.EQ.1)CALL PLOT(0.0,4.5,-3)
     IF(IRUN.EQ.2)CALL PLOT(10.,-4.5,-3)
100  CONTINUE
     GOTO1
999   STOP
      END
SUBROUTINE PRGCIP,IAE)
DIMENSION P(2)
REAL IAE
ND=DEL/H+.5
NH=TS/H+.5
IF(IRUN.EQ.2)NH=0
ND1=TS/(H+2.)+.5
IF(IRUN.EQ.1)ND=ND1
IF(IRUN.EQ.2)ND=ND+ND1
IF(IRUN.EQ.1)HTS=TS
IF(IRUN.EQ.2)HTS=H
ND=ND+1
SUM=0
SP=1.0
T=0.0
I=0
X0=0.0
X1=0.0
U1=0.0
IHOLD=0.0
IAE=0.0
0010 IF (I .LT. ND) G0T011
10 X(I)=0.0
15 I=I+1
T=T+H
TIME(I)=T
E=SP-X(I)
IAE=IAE+ABS(E)*H
20 IF (IHOLD .LT. NHOLD) G0T021
IHOLD=0
SUM=SUM+P(I)*E
U1=SUM/P(I)
21 IHOLD=IHOLD+1
35 X0=X0+H/TAU1*(U1-X0)
X1=X1+H/TAU2*(X0-X1)
IND=I+ND
X(IND)=X1
69 FORMAT(5F20.6)
IF (IAE.LT.1.0E20.AND.I.LT.IMAX) G0T015
IF (I.LT.IMAX) IAE=IAE*(1.0+1.0/T)
RETURN
END
PROGRAM III-3

C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *                                FREQUENCY RESPONSE - DEAD TIME APPROXIMATION *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C * *THIS PROGRAM COMPARES THE FREQUENCY RESPONSE OF A SAMPLER AND ZERO-ORDER HOLD WITH THE RESPONSE OF A PURE DEAD TIME OF ONE-HALF THE SAMPLING TIME. IN BOTH CASES THE OUTPUT IS FILTERED BY A FIRST ORDER LAG. *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *COMPLEX Z,G,GC,S,Z1,Z2
DIMENSION G1(1002),G2(1002),TH1(1002),TH2(1002),WP(1002)
1 BUFFER(5000)
CALL PLOTS(BUFFER,5000)
CALL FPTRAP(-3)
1 READ 10,IOUT,T
 IF(1P.EQ.5)IP=0
10 FORMAT(I1,F9.0)
 IP=0
 L1=0.
 L2=0.
 LL1=0
 LL2=0
 N=1000
 W1=.01
 W2=10.
 DN=N
 DW=(ALOG10(W2)-ALOG10(W1))/DN
 WL= ALOG10(W1)-DW
DO500 I=1,N
WL=WL+DW
W=10.**WL
S=CMPLX(0.0,W)
Z=CMPLX(COS(W*T),SIN(W*T))
Z=CEXP(T*S)
Z1=Z-1.
Z2=Z-EXP(-T)
G=(Z1-1.)/Z2-EXP(-T)/(Z2**2)*Z1
GL=ALOG10(REAL(CABS(G)))
TH=ATAN(REAL(G),IMAG(G))
GC=CEXP(-T*S/2.0)/(S+1.0)**2
GCL=ALOG10(REAL(CABS(GC)))
THC=ATAN(REAL(GC),IMAG(GC))
G2(I)=REAL(CABS(G))
G1(I)=REAL(CABS(G))
TH1(I)=TH
TH2(I)=THC
IF(I.EQ.1)GOTO20
IF(TH1(I).LE.(-270.))L1=1
IF(TH2(I).LE.(-270.))L2=1
IF(TH2(I-1).LE.TH2(I))L2=1
IF(L2.EQ.1.AND.TH1(I-1).LE.TH1(I))L1=1
IF(TH1(I).GT.0.0)TH1(I)=0.0
IF(L1.EQ.1)TH1(I)=-270.
IF(L2.EQ.1)TH2(I)=-270.
WP(I)=W
IP=IP+1
IF(IP.EQ.5)PRINT100,W,G1(I),TH1(I),G2(I),TH2(I)
IF(IP.EQ.5)IP=0
100 FORMAT(1X6F15.6)
IF(G1(I).GT.1.)G1(I)=1.
IF(G2(I).GT.1.)G2(I)=1.
IF(G1(I).LT.01)G1(I)=.01
IF (G2(I) .LT. .01 ) G2(I) = .01

CONTINUE
N01 = N + 1
N02 = N + 2
G1(N01) = .01
G2(N01) = .01
WP(N01) = .01
G1(N02) = 1./3.
G2(N02) = 1./3.
WP(N02) = 1./3.
TH1(N01) = -270.
TH2(N01) = -270.
TH1(N02) = 45./9.
TH2(N02) = 45./9.
CALL FACTOR(1.0)
CALL AXIS(0.0, 0.0, 11, PHASE ANGLE, 11, 3., 90., -270., 90., 10.)
CALL LGAXIS(0.0, 0.0, 9, HFREQUENCY, -9, 5.0, 0.0, 1, 3.0/5.0)
CALL FACTOR(5./9.)
CALL LGAXIINE(WP, TH1, N, 1, 0, 1, -1)
CALL LGAXIINE(WP, TH2, N, 1, 0, 1, -1)
CALL FACTOR(1.0)
CALL PLOT(0.0, 3.8, -3)
CALL LGAXIS(0.0, 0.0, 9, HAMAGNITUDE, 9, 1.0/3, 90., 01, 3./5.)
CALL LGAXIS(0.0, 0.0, 9, HFREQUENCY, -9, 5.0, 0.0, 1, 3./5.)
CALL FACTOR(5./9.)
CALL LGAXIINE(WP, G1, N, 1, 0, 1, 0)
CALL LGAXIINE(WP, G2, N, 1, 0, 1, 0)
CALL FACTOR(1.0)
CALL PLOT(9., -3.8, -3)
IF (IOUT .EQ. 0 ) GOT01
CALL PLOT(0.0, 0.0, 999)

STOP
END
FUNCTION ATAM(Y,X)
A = ATAN(ABS(Y)/ABS(X)) * 180. / 3.14
IF(Y.GT.0.0.AND.X.GT.0.0) ATAM = -360. + A
IF(Y.LT.0.0.AND.X.GT.0.0) ATAM = -A
IF(Y.LT.0.0.AND.X.LT.0.0) ATAM = -180. + A
IF(Y.GT.0.0.AND.X.LT.0.0) ATAM = -180. - A
RETURN
END
PROGRAM IV-1

C **************************************************************************
C TIME OPTIMAL CONTROL OF A HIGH ORDER PROCESS
C **************************************************************************

C THIS PROGRAM INVESTIGATES SEVERAL METHODS FOR IMPLEMENTING A
C PRACTICAL TIME OPTIMAL CONTROL OF A HIGH ORDER OVER DAMPED PROCESS
C

DIMENSION Y(8), YY1(1002), YY2(1002), UU(1002), T(1002), BUFFER(5000)
DIMENSION YX1(1002), YX2(1002)
DIMENSION TAU(7)

COMMON BUFFER
DATA Y/8*-.5/
DATA DELAY/1000*-.5/
CALL PLOTS(BUFFER,5000)
M=.02
IUC=1

1 READ 1000, ISTUDY
DO21=1,7

2 Y(I)=-.5
DO31=1,1000

3 DELAY(I)=-.5
1000 FORMAT(I1)

IF(ISTUDY.EQ.1) T1=3.032
IF(ISTUDY.EQ.2) T1=3.032
IF(ISTUDY.EQ.3) T1=1.850
T2=T1
DEL=2.38
IF(ISTUDY.EQ.4) T1=2.241
IF(I $\text{STUDY}$.EQ.4) T2=2.663
IF(I $\text{STUDY}$.EQ.4) DEL=1.80
IF(I $\text{STUDY}$.EQ.5) T1=4.961
IF(I $\text{STUDY}$.EQ.5) T2=3.803
IF(I $\text{STUDY}$.EQ.5) DEL=1.10
NDEL=DEL/H+.5
U=1.
TM=0.0
RESET=0.0
GAIN=0.0
RATE=0.0
CHECK1=.005
CHECK2=.005
SUM=0.
YMO=-.5
YL=YMO
YM1=YMO
Y71=YMO
CALL AXIS(-3.5,-3.5,2HY1,-2,7.0,0.0,-.7,2,20.)
CALL PLOT(0.0,0.0,3)
CALL SWITCH(T1,T2)
CALL PLOT(0.0,0.0,-3)
CALL PLOT(-2.5,0.0,3)
DOI001=1,1000
TM=TM+H
T(I)=TM
C-----PROCESS
Y(I)=U
DO 10 J=2,7
10 Y(J)=Y(J)+H*(Y(J-1)-Y(J))
C------SMITH PREDICTOR
YMO=YMO+H/T1*(U-YMO)
YM1=YM1+H/T2*(YMO-YM1)
II=NDEL*I
DELAY(II) = YM1
III = 1
IF(III.GT.1000) III = III - 1000
IF(ISTUDY.EQ.1) Y7 = YM1
IF(ISTUDY.EQ.2) Y7 = Y(7)
IF(ISTUDY.GE.3) Y7 = YM1 + Y(7) - DELAY(III)

C------DETERMINING STATE
Y2 = (Y7 - Y1) / H
Y1 = Y7
Z1 = Y1 = 5.0
Z2 = Y2 = 5.0
CALL PLOT(Z1, Z2, 2)
YX1(I) = Y1
YX2(I) = Y2

C------CONTROL CHECK - CONVENTIONAL CONTROLLER
Y72 = (Y(7) - Y71) / H
Y71 = Y(7)
YY1(I) = Y71
YY2(I) = Y72
IF(ISTUDY.EQ.1) YY1(I) = Y1
IF(ISTUDY.EQ.1) YY2(I) = Y2
IF(ABS(Y1).GT.CHECK1) GOTO50
IF(ABS(Y2).GT.CHECK2) GOTO50
SUM = -RESET*Y1 + SUM
U = -1.0*GAIN*Y1 + SUM - RATE*Y2
IF(ABS(U).GT.1.0) U = U / ABS(U)
GOTO90

50 IF(I.EQ.1) U = 1
UL = U
U = Z(Y1, Y2, U)
IF(ABS(UL - U) - .1) 55, 55, 51
51 IF(IUC -12) 54, 52, 52
52 IUC = 0
GOTO55
U=UL
IUC=IUC+1
UU(I)=U
CONTINUE
N1=1001
N2=1002
YY1(N1)=0.0
UU(N1)=0.0
YY1(N2)=1.0/2.5
UU(N2)=1.0/2.5
T(N1)=0.0
T(N2)=2.5
CALL PLOT(10.,0.,0.,-3)
CALL AXIS(0.0,-2.5,4HTIME,-4,8.0,0.0,0.0,2.500,10.)
CALL AXIS(0.0,-2.5,8HRESPONSE,8,5.0,90.,-1.0,2./5.,10.)
CALL LINEIT(YY1,1000,1,0,1)
CALL PLOT(20.,0.0,-3)
ISS=1
IF(ISS.EQ.1)GOTO1
STOP
END
PROGRAM IV-2

SUBROUTINE SWITCH(T1, T2)

C * THIS SUBROUTINE GENERATES AND STORES THE SECOND ORDER SWITCHING CURVE TO BE USED IN THE TIME OPTIMAL STUDY.

C

DIMENSION BUFFER(5000)
COMMON BUFFER
COMMON YY1(500), YY2(500)
U = 1
CALL AXIS(-3.5, -3.5, 2HY2, 2, 7.0, 90., -7., 2, 20.)
CALL PLOT(0.0, 0.0, 3)
DO 10 JJJ = 1, 2
CALL PLOT(0.0, 0.0, 3)
Y10 = 0.
Y20 = 0.
DY2 = 0.
DO 1001 = 1, 500
TI = I - 1
TM = -3.0*(TI/499.)**2
IF(ABS(T1 - T2) GT .001) GOTO 40
ET = EXP(-TM/T1)
Y1 = ET*((Y10 - U)*(1.0 + TM/T1) + TM*Y20) + U
Y2 = ET*(Y20*(1.0 - TM/T1) - (Y10 - U)*TM/T1**2)
GOTO 50

GOTO 40

GOTO 50
\[ C_2 = \frac{U_T}{T_1 - T_2} \]
\[ C_1 = -C_2 - U \]
\[ Y_1 = C_1 \exp(-TM/T_1) + C_2 \exp(-TM/T_2) + U \]
\[ Y_2 = -\frac{C_1 \exp(-TM/T_1)}{T_1} - \frac{C_2 \exp(-TM/T_2)}{T_2} \]

\[ YY_1(I) = Y_1 \]
\[ YY_2(I) = Y_2 \]

\[ \text{IF}(|Y_1| \leq 0.70 \text{ AND } |Y_2| \leq 0.70) \text{CALL PLOT}(5 \times Y_1, 5 \times Y_2, 2) \]

\[ \text{FORMAT}(6F20.5) \]
\[ \text{CONTINUE} \]
\[ \text{CALL PLOT(0.0, 0.0, -3)} \]
\[ U = -1. \]

\[ \text{CONTINUE} \]
\[ \text{RETURN} \]
\[ \text{END} \]
PROGRAM IV-3

C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C TIME OPTIMAL - CONTROLLER
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C * THIS SUBFUNCTION CONTAINS THE LOGIC NECESSARY TO USE THE SWITCHING CURVES GENERATED BY PROGRAM IV-2 TO IMPLEMENT TIME OPTIMAL CONTROL. *
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *
C FUNCTION Z(Y1,Y,U)
DIMENSION BUFFER(5000)
COMMON BUFFER
COMMON YY1(500),YY2(500)
Y2=ABS(Y)
ZZ=YY1(1)
ISTRT=1
DO225 I=ISTRT,500
J=I
225 IF(YY2(J).GT.Y2)GOTO226
226 IF(J.EQ.1)GOTO250
230 ZZ=YY1(J)+(YY1(J)-YY1(J-1))/(YY2(J)-YY2(J-1))*(Y2-YY2(J))
250 Z=Y/Y2*ZZ
ZZ=0.0
IF(ABS(U).GT.1)ZZ=U/ABS(U)
IF(Y1.LE.0.AND.Y.GE.0.AND.Z-Y1.LE.0.)ZZ=-1.
IF(Y1.LE.0.AND.Y.GE.0.AND.Z-Y1.GE.0.)ZZ=-1.
IF(Y1.GE.0.AND.Y.LE.0.AND.Z-Y1.GE.0.)ZZ=+1.
IF(Y1.GE.0.AND.Y.LE.0.AND.Z-Y1.LE.0.)ZZ=+1.
Z=ZZ
RETURN
PROGRAM IV-4

TIME OPTIMAL - CURVE FIT

COMMON IDUM(100), DUMM(1100)
DIMENSION TM(2002), XT(2002), XA(2002), U(2), X(8), XL(3), UL(3), Q(3),
STEP(3), BUFFER(5000)
DIMENSION TAU(7)
CALL FPTRAP(-3)
COMMON N, DT, XA, XT, U2
COMMON TM
COMMON JJJJ
JJJJ=0
CALL PLOTS(BUFFER, 5000)
DATA X/0.0, 6*-1.0, 0.*/
DATA TAU, Q/7*1.0, 1.841, 1.863, 2.38/3
Q(1)=2.241
Q(2)=2.663
Q(3)=1.80
1 CONTINUE
U2=0.0
DT=.01
N=2000
T=0.0
DO 30 J = 1, N
T = T + DT
TM (J) = T
DO 10 I = 2, 7
X(I) = X(I) + DT / TAU(I) * (X(I-1) - X(I))
XA(J) = X(7)
30 CONTINUE
XA(2001) = - .1
XA(2002) = .025
XT (2001) = - .1
XT (2002) = .025
TM(2001) = 0.0
TM(2002) = 2.5
DO 40 J = 1, 2
XL(J) = DT * .005
UL(J) = 10000.
40 STEP(J) = .1
XL(3) = 0.0
UL(3) = 4.0
STEP(3) = .1
CALL PATTERN(3, XL, UL, Q, 2, STEP, PHI)
PRINT 32, Q
CALL PROC(NXL, Q, CFUN)
32 FORMAT (1H 10H ANSWERS , 5E20.6)
CALL AXIS(0.0, 0.0, 4H TIME, - 4.8, 0.0, 0.0, 0.0, 2.5, 10.)
CALL AXIS(0.0, 0.0, 8H RESPONSE, 8.5, 0.90, -.1, .025, 10.)
CALL LINE(TM, XA, 2000 , 1, 0, 1)
CALL LINE(TM, XT, 2000 , 1, 0, 1)
CALL PLOT(10., 0.0, -3)
PAUSE
Q(1) = 4.0
Q(2) = 4.0
Q(3) = 0.
GOTO 1

STOP
END
SUBROUTINE PROC(NXL,QO,CFUN)
DIMENSION TM(2002)
DIMENSION Q(3),QQ(3)
COMMON IDUM(100),DUMM(1100)
COMMON N,DT,XA,XT,U
COMMON TM
COMMON J
ER=0
X1=-.1
X2=-.1
DO10 I=1,3
10 Q(I)=QQ(I)
NDEL=Q(3)/DT+1.5
CALL SSWHCH(4,NDELO)
IF(NDELO.EQ.1.AND.NDEL.GT.2)GOTO99
DO11 I=1,NDEL
XT(I)=-.1
11 CONTINUE
DO20 I=NDEL,N
X1=X1+DT/Q(1)*(U-X1)
X2=X2+DT/Q(2)*(X1-X2)
20 XT(I)=X2
DO30 I=1,250
ERR=(XT(I)-XA(I))**2
CALL SSWHCH(3,NDELI)
69 FORMAT(1X3F20.5)
30 ER=ER+(XT(I)-XA(I))**2
CFUN=ER
IF(NDELI.EQ.2)GOTO99
J=J+1
PRINT696,J,Q,ER
696  FORMAT(///1X15,10X,4E15.5)
    CALL LINE(TM,XT,2000,1,0,1)
99  RETURN
    END
PROGRAM V-1

C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
C * * FAST REGRESSION OF FILTERED PULSE DATA * * 
C * * 
C * * THIS PROGRAM DIRECTS THE STUDY OF FORMULATING THE NONLINEAR LEAST-
C * SQUARE FOR FAST ON-LINE ANALYSIS. IT INITIALIZES THE NECESSARY 
C * PARAMETERS REQUIRED BY EACH METHOD OF FORMULATION AND CALLS THE 
C * APPROPRIATE SUBROUTINE. 
C * 
C * 
C * ---PROGRAM TO PERFORM A FAST REGRESSION OF FILTERED PULSE DATA 
C EXTERNAL PROC1,PROC2,PROC3 
C COMMON NP,NT,NORDER,TS,DEL,TFIL,TAU(10),X(1000),T(1000),U(1000) 
C COMMON XX(3,3),UU(3,3),SX(1000) 
C COMMON LCOUNT 
C COMMON NDELR 
C CALL FPTRAP(-3) 
C DIMENSION P(3),STEP(3) 
C DATA STEP/3*.01/ 
C DIMENSION PP(3) 
1 IPROC=0 
READ1000,NORDER,DEL,TFIL,(TAU(I),I=1,NORDER) 
PRINT1000,NORDER,DEL,TFIL,(TAU(I),I=1,NORDER) 
1000 FORMAT(1H1,I4,14F5.2) 
READ1001,TS,TPULSE,TMAX,PP 
PRINT1001,TS,TPULSE,TMAX,PP 
1001 FORMAT(6F10.3) 
NP=TPULSE/TS+.5 
NT=TMAX/TS+.5
CALL DATA
2  IPROC=IPROC+1
  DO3I=1,2
3  P(I)=PP(I)
  LCOUNT=0
  U2=1.
  CALL CLOCK(IT1)
  IF(IPROC.GT.1)GOTO60
C-----OUTPUT CORRELATION TERMS(X(I)+X(I-1)+X(I-2))**2
  DO10J=1,3
10  XX(1,J)=0.0
  DO20I=1,NT
  DO15J=1,3
  IJ=I-J+1
  IF(IJ.LE.0)GOTO15
  XX(1,J)=XX(1,J)+X(I)*X(IJ)
15  CONTINUE
20  CONTINUE
  XX(2,2)=XX(1,1)-X(NT)**2
  XX(2,3)=XX(1,2)-X(NT)*X(NT-1)
  XX(3,3)=XX(2,2)-X(NT-1)**2
C-----INPUT CORRELATION TERMS(U(I)+U(I-1)+U(I-2))**2
  XNP=NP
  DO30I=1,3
  DO30J=1,3
  XJI=J-I
30  UU(I,J)=(XNP-XJI)*U2
  DO40I=1,3
  DO40J=I,3
  XX(J,I)=XX(I,J)
40  UU(J,I)=UU(I,J)
C-----COMPUTING CROSS PRODUCT TERMS AS A FUNCTION OF DEAD TIME
  SX(I)=0.0
  NP1=NP-1
DO50I=1,NP1
50 SX(I)=SX(I)+X(I)
    NTNP=NT-NP
DO60I=1,NTNP
    INP=I+NP-1
    I1=I-1
    SX(I+1)=SX(I)+X(INP)
    IF(I1.GE.1)SX(I+1)=SX(I+1)-X(I)
    CONTINUE
    CALL CLOCK(IT2)
    ITEST=0
    NDELR=-100
    IF(IPROC.EQ.1)CALL PATERN3,P,STEP,1,CFUN,PROC1
    IF(IPROC.EQ.2)CALL PATERN3,P,STEP,1,CFUN,PROC2
    IF(IPROC.EQ.3)CALL PATERN3,P,STEP,1,CFUN,PROC3
    CALL CLOCK(IT3)
    PRINT1007,LCOUNT,CFUN
1007 FORMAT(1H013HANSWERS AFTER ,I3,2X,23HFUNCTIONAL EVALUATIONS //
    1 5X5HCOST=,E15.6,20X,18HOPTIMAL PARAMETERS )
    PRINT1008,(J,P(J),J=1,3)
1008 FORMAT(5X,5(I10.F10.3))
    TIME1=IT2-IT1
    TIME2=IT3-IT1
    TIME3=IT3-IT2
    TIME1=TIME1/1000.
    TIME2=TIME2/1000.
    TIME3=TIME3/1000.
    COUNT=LCOUNT
    TIME4=TIME3/COUNT
    PRINT5000,TIME2,TIME1,TIME3,TIME4
5000 FORMAT(/1X25HEXECUTION TIME-TOTAL = ,F10.3,7HSECONDS/}
    1 1X25HEXECUTION TIME-MAIN = ,F10.3,/
    1 1X25HEXECUTION TIME-SEARCH = ,F10.3,/
    1 1X25HAVERAGE TIME/EVALUATION= ,F10.3)
IF(IPROC.GE.3)GOTO1
GOTO2
999 CONTINUE
STOP
END
PROGRAM V-2

SUBROUTINE DATA

C * *
C * FAST REGRESSION - PROCESS DATA
C * *
C * THIS SUBROUTINE GENERATES AND FILTERS THE DATA USED IN THE FAST
C * REGRESSION STUDY.
C * *
C
COMMON NP,NT,NORDER,TS,DEL,TFIL,TAU(10),X(1000),T(1000),U(1000)
COMMON XX(3,3),UU(3,3),SX(1000)
COMMON LCOUNT
DIMENSION XY(11)
H=.01
IF(TFIL.LT.H)TFIL=H
NTS=TS/H+.5
NPP=NTS*NP
X(11)=0.0
XF=0.0
T(11)=0.0
SP=0.0
DO111=1,11
11 XY(11)=0.0
TI=0.0
NDEL=DEL/H+.5
IF(NDEL.LE.0)SP=1.0
U(11)=SP
N2=NORDER+1
III=1
II=0.0
I=1
200  I=I+1
  TI=TI+H
205  XF=XF+(SP-XF)*H/TFIL
  IF(I.EQ.NDEL+1) SP=1.0
  IF(I.EQ.(NDEL+NPP+1)) SP=0.0
  XY(1)=XF
  DO210 J=2,N2
210  XY(J)=XY(J-1)+(XY(J-1)-XY(J))*H/TAU(J-1)
  II=II+1
  IF(II.LT.NTS) GOTO200
  II=0
  III=III+1
  U(III)=SP
  X(III)=X(N2)
  T(III)=TI
  IF(III.LT.NT) GOTO200
  IDEL=0
  DO250 I=1,NT
  IF(ABS(U(I)).GT.0.01) GOTO300
250  IDEL=IDEL+1
300  DO350 I=1,NT
  ID=IDEL+1
  IF(ID.LT.NT) U(ID)=U(I)
  IF(ID.GE.NT) U(ID)=0.0
350  CONTINUE
999  RETURN
END
PROGRAM V-3

SUBROUTINE PROCI(P,CFUN)

C * * * FAST REGRESSION - SIMPLIFIED FORMULATION *
C * * * THIS SUBROUTINE CONTAINS THE SIMPLIFIED FORMULATION OF THE NON-LINEAR *
C * REGRESSION. *
C *
C COMMON N P,N T,NORDER,TS,DEL,TFIL,TAU(1O),X (1000),T (1000),U (1000)
COMMON X X(3,3),UU(3,3),SX(1000)
COMMON LCOUNT
COMMON NDELR
DIMENSION C(6),XU(3,3),P(3)
DOUBLE PRECISION CI,CJ,Q(6,6),ERR,CICJ

C---- COMPUTING CROSS CORRELATION TERMS FOR DEAT TIME, NDELR
LCOUNT=LCOUNT+1
NDELR=P(3)/TS
IF(NDELR.EQ.NDELR)GOTO21
DO20I=1,3
DO10J=1,3
JI=J-I+NDELR+3
10 XU(I,J)=SX(JI)
20 CONTINUE
21 CONTINUE
NDELR=NDELR

C---- CONSTANTS WHICH SPECIFY RESPONSE, C(6)
X N=NDELR
X DEL=P(3)/TS-X N
A = EXP(-TS/P(2))
B = EXP(-TS/TFIL)
AM = A ** (1.0 - XDEL) / TFIL
BM = B ** (1.0 - XDEL) / P(2)
D = 1.0 / P(2) - 1.0 / TFIL
G = P(1) / D
C(1) = 1.0
C(2) = -(A + B)
C(3) = A * B
C(4) = -G * (D + AM - BM)
C(5) = G * (D * (A + B) + AM * (1.0 + B) - BM * (1.0 + A))
C(6) = -G * (D * A * B + AM * B - BM * A)

C----DETERMINING ERRDR
DO50 I = 1, 6
DO40 J = 1, 6
IF (I .LE. 3. AND. J .LE. 3) Q(I, J) = XX(I, J)
IF (I .GT. 3. AND. J .GT. 3) Q(I, J) = UU(I - 3, J - 3)
IF (I .LE. 3. AND. J .GT. 3) Q(I, J) = XU(I, J - 3)
CI = C(I)
CJ = C(J)
Q(I, J) = Q(I, J) * CI * CJ
40 Q(J, I) = Q(I, J)
50 CONTINUE
ERR = 0.0
DO60 I = 1, 6
DO60 J = 1, 6
60 ERR = ERR + Q(I, J)
999 CONTINUE
CFUN = ERR
RETURN
END
SUBROUTINE PROC2(P, CFUN)

* FAST REGRESSION - CONVENTIONAL FORMULATION 1 *

* THIS SUBROUTINE CONTAINS THE CONVENTIONAL FORMULATION IN WHICH THE *
* UNEXPANDED ERROR TERM IS EVALUATED EACH ITERATION. *

COMMON NP, NT, NORDER, TS, DEL, TFIL, TAU(10), X(1000), T(1000), U(1000)
COMMON XX(3,3), XU(3,3), SX(1000)
COMMON LCOUNT
COMMON I TEST
DIMENSION C(6), XU(3,3), P(3)
LCOUNT = LCOUNT + 1
NDEL = P(3)/TS
XN = NDEL
XDEL = P(3)/TS - XN
A = EXP(-TS/P(2))
B = EXP(-TS/TFIL)
AM = A**(1.-XDEL)/TFIL
BM = B**(1.-XDEL)/P(2)
D = 1./P(2) - 1./TFIL
G = P(1)/D
C(1) = 1.0
C(2) = -(A+B)
C(3) = A*B
C(4) = -G*(D+AM-BM)
C(6) = -G*(D*A*B + A*M*B - B*M*A)
CFUN = 0.0
X1 = 0.
X2 = 0.
U1 = 0.
U2 = 0.
U3 = 0.
D075K = 1, NT
X0 = X(K)
IF(K . GT. 1) X1 = X(K - 1)
IF(K . GT. 2) X2 = X(K - 2)
KN = K - NDEL
IF(KN . GT. 1) U1 = U(KN - 1)
IF(KN . GT. 2) U2 = U(KN - 2)
IF(KN . GT. 3) U3 = U(KN - 3)
XM = -(X1*C(2) + X2*C(3) + U1*C(4) + U2*C(5) + U3*C(6))
CFUN = CFUN + (X0 - XM)**2
75 CONTINUE
RETURN
END
PROGRAM V-4

SUBROUTINE PROC3(P, CFUN)

        FAST REGRESSION - CONVENTIONAL FORMULATION 2

        This subroutine contains the conventional formulation in which each summation term in the expanded fit-error equation is evaluated.

        Subroutine to calculate error by evaluating all the sum terms of the expansion of the error expression

        COMMON NP, NT, NORDER, TS, DEL, TFIL, TAU(10), X(1000), T(1000), U(1000)
        COMMON XX(3, 3), UU(3, 3), SX(1000)
        COMMON LCOUNT
        DIMENSION C(6), XU(3, 3), P(3)
        DIMENSION CC(6), QQ(6, 6)
        DOUBLE PRECISION CI, CJ, Q(6, 6), ERR, CICJ
        LCOUNT = LCOUNT + 1
        NDEL = P(3)/TS
        XN = NDEL
        XDEL = P(3)/TS - XN
        A = EXP(-TS/PI2)
        B = EXP(-TS/TFIL)
        AM = A**(1. - XDEL)/TFIL
        BM = B**(1. - XDEL)/P(2)
        D = 1./P(2) - 1./TFIL
        G = P(1)/D
        C(1) = 1.0
        C(2) = -(A+B)
CEUN=ERR
RETURN
END
PROGRAM VI-1
C
C  CONTINUOUS LEAST-SQUARE REGRESSION STUDY - MAIN PROGRAM
C
C  THIS PROGRAM DIRECTS THE STUDY OF THE FORMULATION OF THE NON-LINEAR
C  LEAST-SQUARE REGRESSION FOR CONTINUOUS ON-LINE ANALYSIS.
C
C
COMMON XX1,XX00,XX01,XXU(50),XXU(50),UU(50),UU(50),F1,F2
COMMON 10,IMAX,IPROC,IRUN,H,TS,TMAX,TO
COMMON TIME(1000),X(1000),U(1000)
COMMON AA(50),BB(50),EE(50)
DIMENSION TM(1000),GM(1000),DEL(1000)
1  READ1000,IPLOT,IPROC,IRUN,TF,TMAX
  IF(IPLOT.EQ.9)GOT0999
1000  FORMAT(2I1,12,1X,3F5.2)
    TO=TF
    PRINT1001,IPLOT,IPROC,IRUN,TF,TMAX
1001  FORMAT(1H1,2I1,12,1X,3F5.2)
    CALL FPTRAP(-3)
    H=.02
    TS=.1
    ITS=TS/H+.5
    TS=ITS
    TS=TS*H
    IMAX=TMAX/TS
    IO=(TO+.5)/TS+.5
    IF(IPROC.EQ.1)CALL DATA1
10  CONTINUE
GM(1)=1.0
TM(1)=1.0
DTMAX=.05
DGMAX=.05
X1=0.0
XX1=0.0
XXO=0.0
XXO1=0.0
DO50I=1,50
UZ(I)=0.0
X1U(I)=0.0
XOU(I)=0.0

50  UU(I)=0.0
NMIN=0
F1=EXP(-TS/TF)
F2=1.0-F1
DO40I=1,50
AA(I)=EXP(-TS/1.0)
BB(I)=1.0-AA(I)
EE(I)=10.
IF(I.EQ.7)EE(I)=0.0

40  CONTINUE
IOIM=IO+1MAX
DO100I=1,IOIM
CALL FILTR(X1,X(I),U(I))
M=I
IF(I.GT.1)TM(I)=TM(I-1)
IF(I.GT.1)GM(I)=GM(I-1)
IF(M.LT.1)GOTO100
CALL PAR(NMIN,A,B,E)
IF(A.LE.0.0)GOTO102

55  TM1=-TS/ALOG(A)
GM1=B/(1.-A)
DT=TM1-TM(I)
DG = GM1 - GM(I)
IF (ABS(DT).GT. DTMAX) DT = DT/ABS(DT) * DTMAX
IF (ABS(DG).GT. DGMAX) DG = DG/ABS(DG) * DGMAX
TM(I) = DT + TM(I)
GM(I) = DG + GM(I)
D = NMIN
DELM(I) = D * TS
102 CONTINUE
CALL SSWSWHCH(I,KK)
IF (KK.EQ.1) PRINT69, TIME(I), X(I), U(I), TM(I), GM(I), DELM(I), E
69 FORMAT (1X7E15.5)
100 CONTINUE
DIMENSION BUFFER(5000)
CALL PLOTS(BUFFER, 5000)
CALL FACTOR(8./10.)
ZM1 = 4.0
ZM2 = 4.0
ZM3 = 4.0
ZM4 = 4.0
DO 110 I = 1, IMAX
110 I = I + 10
DELM(I) = DELM(IOI)
TM(I) = TM(IOI)
GM(I) = GM(IOI)
X(I) = X(IOI)
U(I) = U(IOI)
TIME(I) = TIME(IOI)
CALL SSWSWHCH(I, KK)
IF (KK.EQ.1) PRINT69, TIME(I), X(I), U(I), TM(I), GM(I), DELM(I), E
IF (ABS(DELM(I)).GT. ZM1) DELM(I) = DELM(I)/ABS(Delm(I)) * ZM1
IF (ABS(TM(I)).GT. ZM2) TM(I) = TM(I)/ABS(TM(I)) * ZM2
IF (ABS(GM(I)).GT. ZM3) GM(I) = GM(I)/ABS(GM(I)) * ZM3
IF (ABS(X(I)).GT. ZM4) X(I) = X(I)/ABS(X(I)) * ZM4
IF (ABS(U(I)).GT. ZM4) U(I) = U(I)/ABS(U(I)) * ZM4
CONTINUE
TM(IMAX+2)=2.0
GM(IMAX+2)=2.0
DELM(IMAX+2)=2.0
X(IMAX+2)=2.0
U(IMAX+2)=2.0
DELM(IMAX+1)=-8.0
GM(IMAX+1)=-5.5
TM(IMAX+1)=-3.5
X(IMAX+1)=-2.5
U(IMAX+1)=-1.25
TIME(IMAX+2)=TMAX/10.
TIME(IMAX+1)=TIME(1)
IF(IPILOT.EQ.1)GOTO101
CALL PLOT(18.,0.,0.,-3)
CALL AXIS(0.0,0.0,40,TIME,-4,10,,0.0,0.0,TMAX/10.,10.)
CALL PLOT(0.00,0.00,3)
CALL PLOT(0.00,1.75,2)
CALL PLOT(0.00,2.00,3)
CALL PLOT(0.00,2.75,2)
CALL PLOT(0.00,3.00,3)
CALL PLOT(0.00,3.75,2)
CALL PLOT(0.00,4.00,3)
CALL PLOT(0.00,4.75,2)
CALL PLOT(0.00,4.00,3)
CALL PLOT(0.00,4.00,2)
CALL PLOT(0.00,3.00,3)
CALL PLOT(0.00,3.00,2)
CALL PLOT(0.00,2.00,3)
CALL PLOT(10.0,4.00,2)
CALL PLOT(10.0,3.00,3)
CALL PLOT(0.00,2.00,3)
CALL PLOT(10.0,2.00,2)
CALL PLOT(10.0,2.00,3)
CALL LINE(TIME,X,IMAX,1,0,1)
T2=1./TIME(IMAX+2)
U1=U(IMAX+1)
U2 = 1.0/(IMAX + 2)
CALL PLOT(0.0, -U1*U2, 3)
IM1 = IMAX - 1
DO 210 I = 1, IM1
   CALL PLOT(TIME(I)*T2,(U(I) - U1)*U2, 2)
210 CALL PLOT(TIME(I+1)*T2,(U(I) - U1)*U2, 2).
CONTINUE
CALL LINE(TIME, TM, IMAX, 1, 0, 1)
CALL LINE(TIME, GM, IMAX, 1, 0, 1)
CALL LINE(TIME, DELM, IMAX, 1, 0, 1)
CALL PLOT(0.0, 0.0, -3)
GOTO 1
999 CALL PLOT(0.0, 0.0, 999)
STOP
END
SUBROUTINE FILTR(X1, X0, UO)

CONTINUOUS LEAST-SQUARE - FILTER

* THIS SUBROUTINE MAINTAINS A RUNNING ESTIMATE OF THE PRODUCT AND CROSS-
* PRODUCT TERMS BY WAY OF A FIRST ORDER FILTER.

COMMON XX11, XX00, XX01, XOU(50), X1U(50), UU(50), U(50), F1, F2
XX11 = XX00
XX00 = XX00 + F1 + X0 * X0 * F2
XX01 = XX01 + F1 + X0 * X1 * F2
DO10 I = 1, 24
J = 26 - I
U(J) = U(J-1)
10
UU(J) = UU(J-1)
UU(I) = UU(I) + U0 * U0 * F2
U(I) = U0
DO20 I = 1, 25
XOU(I) = XOU(I) + F1 + X0 * U(I) * F2
X1U(I) = X1U(I) + F1 + X1 * U(I) * F2
20 CONTINUE
X1 = X0
RETURN
END
PROGRAM VI-3

SUBROUTINE PAR(NMIN, A1, B1, E1)

CONTINUOUS LEAST-SQUARE PARAMETER CALCULATIONS

THIS SUBROUTINE MAINTAINS AN RUNNING ESTIMATE OF THE PROCESS
PARAMETERS BASED ON THE PRODUCT AND CROSS PRODUCT TERMS CALCULATED BY PROGAM VI-2

COMMON XX11,XX00,XX01,XXU(50),XX1U(50),UU(50),UZ(50),F1,F2
COMMON I0,IMAX,IPROC(IRUN,H,TS,TMAX,TO
COMMON TIME(1000),X(1000),U(1000)
COMMON AA(50),BB(50),EE(50)
CALL SSWTCH(KK)

NN=1
DO100 N=1,25
DEN=UU(N)*XX11-XXU(N)**2
IF(DEN.LT.1.0E-6 )GOTO50
AA(N)=(UU(N)*XX01-XXU(N)*XXU(N))/DEN
A=AA(N)
BB(N)=(XXU(N)-A*XXU(N))/UU(N)
B=BB(N)
EE(N)=XX00-2.0*A*XX01-2.0*B*XXU(N)+A*A*XX11+2.0*A*B*XXU(N)+B*3*UU(N)
50 CONTINUE
IF(EE(N).LE.EE(NN))NN=N
IF(KK.EQ.2)GOTO100
PRINT69,N,NN,EE(N),DEN
69 FORMAT(1X215,2E20.8)
100 CONTINUE
A1 = AA(NN)
B1 = BB(NN)
E1 = EE(NN)
NMIN = NN - 2
40 RETURN
END
PROGRAM VI-4

SUBROUTINE DATA1

C *CONTINUOUS LEAST-SQUARE - GENERAL PROCESS*

C *THIS SUBROUTINE CONTAINS A SIMULATION OF A FIRST ORDER LAG PLUS DEAD TIME WITH SEVERAL OPTIONS OF BOTH CONTROLLED AND UNCONTROLLED INPUTS TO EVALUATE THE PERFORMANCE OF THE REGRESSION UNDER VARIOUS CONDITIONS.*

COMMON XX11,XX00,XX01,X0U(50),X1U(50),UU(50),UZ(50),F1,F2
COMMON IO,IMAX,IPROC,IRUN,H,TS,TMAX,T0
COMMON TIME(1000),X(1000),U(1000)
DIMENSION Z(2000)
A1=-10.
A2=-20.
R=RAND(A1,A2)
A1=A2
ITO=0
P1=1.17
P2=1.01
G1=EXP(-H)
G2=(1.-G1)
NOEL= .5/H+.5
NOEL= 25
NOEL1=25
IST=TS/H+.5
U0=0.0
T = -1.0E-10
X0 = 0.0
SP = 0.0
D = 0.0
Y = 0.0
GAIN = 1.0
SUM = 0.0
TT = -10
TT = TT*TS
DOSI = 1,500
5
Z(I) = 0.0
I = 1
J = 0
IR = 0
L = 0
C----- CONTROLLER
9 CONTINUE
CALL SSWTCH(4, K4)
IF(K4.EQ.1) PRINT 1000, T, XO, U0, SP
1000 FORMAT (1X4F20.5)
U0 = 0.0
IF(IR.EQ.1) Z(I) = RAND(A1, A2)/60. + Z(I)
X0 = Z(I)
E = SP - XO
SUM = SUM + E*TS
C----- RUN OPTIONS
GOTO (110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220), IRUN
C----- OPEN LOOP PULSE
110 IF(T.GE..0) U0 = 1.0
IF(T.GE.5.) U0 = 0.0
GOTO 60
C----- CLOSED LOOP SET-POINT CHANGE
120 U0 = P1*E + P2*SUM
IF(T.GE..0) SP = 1.0
GOTO60
C----CLOSED LOOP LOAD CHANGE
130 U0=P1*E+P2*SUM
   IF(T.GE.0.0)D=1
   GOTO60
C----OPEN LOOP NOISE
140 U0=RAND(A1,A2)/3.
   GOTO60
C----CLOSED LOOP NOISE -- SET-POINT CHANGE
150 U0=P1*E+P2*SUM
   SP=RAND(A1,A2)/6.0
   GOTO60
C----CLOSED LOOP NOISE -- LOAD CHANGE
160 U0=P1*E+P2*SUM
   D=RAND(A1,A2)/6.
   GOTO60
C----CLOSED LOOP MEASUREMENT NOISE
170 U0=P1*E+P2*SUM
   IR=1
   GOTO60
C----OPEN LOOP NOISE -- LAG CHANGE
180 IF(T.GE.0.0)G1=EXP(-H/2.0)
    IF(T.GE.0.0)G2=(1.-G1)
    U0=RAND(A1,A2)/3.
    GOTO60
C----OPEN LOOP NOISE -- GAIN CHANGE
190 IF(T.GT.0.0)GAIN=2.0
    U0=RAND(A1,A2)/3.
    GOTO60
C----OPEN LOOP NOISE -- DEAD TIME CHANGE
200 IF(T.GE.0.0)ND1=50
    U0=RAND(A1,A2)/3.
    GOTO60
C----CLOSED LOOP -- SET-POINT CHANGE -- NOISE IN LOAD
210 IF(T.GE.0.0)SP=1.0
    D=RAND(A1,A2)/6.0
    UO=P1*E+P2*SUM
    GOTO60

C----CLOSED LOOP -- SET-POINT CHANGE -- MEASUREMENT NOISE
220 IF(T.GE.0.0)SP=1.0
    UO=P1*E+P2*SUM
    IR=1
    GOTO60
60 ITO=ITO+1
    IF(ITO.LE.5)UO=UO+10.
61 J=J+1
    IF(ITO.GE.10)T=T+TS
    TT=TT+TS
    TIME(J)=TT
    U(J)=UO
    X(J)=XO

C----PROCESS
70 Y=Y*G1+(UO+D)*G2*GAIN
    IF(NDEL.EQ.NDEL1)GOTO80
    D075K=1,25
    NDEL1=K+NDEL+I-1
75 Z(INDEL1)=Y
    NDEL=NDEL1
80 INDEL=I+NDEL
    IF(INDEL.GT.2000)INDEL=INDEL-2000
    Z(INDEL)=Y
    I=I+1
    IF(I.GT.2000)I=1
    L=L+1
    IF(L.LT.IST)GOTO70
    L=Q
    IF(J.LT.IMAX+10+5)GOTO9
RETURN
PROGRAM VI-5

SUBROUTINE DATA2(WAM)

C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C *
C * CONTINUOUS LEAST-SQUARE - REACTOR
C *
C *
C *
C * THIS SUBROUTINE CONTAINS A SIMULATION OF A NON-IDEAL PLUG FLOW
C * REACTOR TO EVALUATE THE CONTINUOUS LEAST-SQUARE REGRESSION ON A
C * REALISTIC PROCESS.
C *
C *
C DIMENSION P(2)
COMMON XX11,XX00,XX01,XOU(50),X1U(50),UU(50),U2(50),F1,F2
COMMON IO,IMAX,IPROC,IRUN,H,ST,TMAX,TO
COMMON TIME(2500),X(2500),U(2500)
REAL MASS1,MASS2,K0
REAL KM
DIMENSION TR(11),TS(11),CA(11),CB(11),CR(11),RATE(11)
C-----REACTOR CONSTANTS
VR=.400
VS=.380
S=3.468
WS=48.
WRO=18.
WA=0.0
CAO=0.0
CBO=58.0
CRO=2.0
MASS1=60.
MASS2=62.4
KO=175.
EO=500.
HC=50.\*60.
HV=500.
CP1=1.0
CP2=1.0
DATA CA/ 0.000, 0.000, 0.000, 0.000, 0.000, 0.000/
1 0.000, 0.000, 0.000, 0.000, 0.000, 0.000/
DATA CB/ 58.000, 58.000, 58.000, 58.000, 58.000, 0.000/
1 58.000, 58.000, 58.000, 58.000, 58.000, 58.000/
DATA CR/ 2.000, 2.000, 2.000, 2.000, 2.000, 2.000/
1 2.000, 2.000, 2.000, 2.000, 2.000, 2.000/
DATA TR/ 70.000, 59.757, 59.031, 58.979, 58.976, 58.000/
1 58.975, 58.975, 58.975, 58.975, 58.975, 58.975/
DATA TS/ 55.000, 58.693, 58.975, 58.975, 58.975, 58.975/
1 58.975, 58.975, 58.975, 58.975, 58.975, 58.975/
USTRT = 0.0000
XSTRT=2.0
SP00=5.
TS(1)=55.
RATE(1)=0.0
TPULSE=TMAX/12.
TP=TMAX
NPS=ST/H+.5
HTS=ST
SP0=0.0
P(1)=.3
P(2)=1.0
U0=0.0
J=0
L=0
T=-.5
IF(IRUN.EQ.2)T=-1.0
SUM=0.0
100 DC200 II=1,NPS
  L=L+1
  T=T+H
  X11=II
  WA=U0+USTRT
  IF(IRUN.EQ.1)WA=WAM
  IF(IRUN.EQ.1 .AND. T.LT.0.0)WA=WAM-1.
  WR=WA+WRO
  IF(WA.GE.12.)WA=12.
  IF(WA.LE.0.0)WA=0.0
  CB(1)=WRO/WR*CBO
  CR(1)=WRO/WR*CR0
  CA(1)=60.-CB(1)-CR(1)
  FR2=EXP(-H/(VR/WR))
  FR1=1.0-FR2
  FS2=EXP(-H/(VS/WS))
  FS1=1.0-FS2
  DD150 I=2,11
C------MATERIAL BALANCE - REACTOR
  RATE(I)=VR*CA(I)*CB(I)*KO*EXP(-EO/TR(I))
  AI =FR1*(CA(I-1)-RATE(I)/WR)+FR2*CA(I)
  CB(I)=FR1*(CB(I-1)-RATE(I)/WR)+FR2*CB(I)
  CA(I)=AI
  IF(CA(I).LT.0.0)CA(I)=0.0
  IF(CB(I).LT.0.0)CB(I)=0.0
  CR(I)=MASS1-CA(I)-CB(I)
C------ENERGY BALANCE - REACTOR
  TR(I)=FR1*(TR(I-1)+(RATE(I)*HV-HC*S*(TR(I)-TS(I)))/(MASS1*WR*CP1))
150   +FR2*TR(I)
C------ENERGY BALANCE - SHELL
  TS(I)=FS1*(TS(I-1)+HC*S*(TR(I)-TS(I)))/(MASS2*WS*CP2)+FS2*TS(I)
  CONTINUE
  XO =CR(11)-XSTRT
  CONTINUE
IF(IRUN.EQ.2)GOTO49
IF(IRUN.EQ.3)GOTO48
GOTO50
48
   UO=10.
   IF(T.LT.0.0 .OR. T.GE.0.1)UO=0.0
   GOTO199
49
   IF(T.LT.0.0)UO=0.0
   IF(T.LT.(-8))UO=.5
   IF(T.LT.0.0)GOTO199
   TP=TP+ST
   IF(TP.LT.TPULSE)GOTO199
   TP=0.0
   UO=UO+1.
   GOTO199
C------PROPORTIONAL + INTEGRAL CONVENTIONAL
50
   SUM=SUM+(SPO-XO)*HTS
   UO=P(1)*(SPO-XO)+P(2)*SUM
199
   CONTINUE
   IF(UO+USTRT.GE.12.)UO=12.-USTRT
   IF(UO+USTRT.LE.0.0)UO=-USTRT
   CALL SSWTCH(3,KK)
   IF(KK.EQ.1)PRINT6900,(K,CA(K),CB(K),CR(K),TR(K),TS(K),K=1,11)
6900
   FORMAT(11(1X,115, 5F15.8/,1H0))
   IF(IRUN.EQ.2)GOTO70
   IF(T.LT.0.0)GOTO100
70
   J=J+1
   U(J)=UO
   X(J)=XO
   TIME(J)=T
69
   FORMAT(1X3F20.8)
   IF(T.LT.TMAX)GOTO100
   IMAX=J
999
   CONTINUE
7001
   FORMAT(6X8HDATA CA/,5(F8.3,1H,),/5X1H1,8X5(F8.3,1H,),F8.3,1H/)
7002 FORMAT(6X8HDATA CB/,5(F8.3,1H),/5X1H1,8X5(F8.3,1H),F8.3,1H/)
7003 FORMAT(6X8HDATA CR/,5(F8.3,1H),/5X1H1,8X5(F8.3,1H),F8.3,1H/)
7004 FORMAT(6X8HDATA TR/,5(F8.3,1H),/5X1H1,8X5(F8.3,1H),F9.3,1H/)
7005 FORMAT(6X8HDATA TS/,5(F8.3,1H),/5X1H1,8X5(F8.3,1H),F8.3,1H/)
7006 FORMAT(6X6HUSTRT=,F15.4)
68 FORMAT(1XE20.5,2F15.5)
RETURN
END
PROGRAM VII-1

C * PREDICTOR ALGORITHM STUDY - MAIN PROGRAM
C *
C * THIS PROGRAM DIRECTS THE STUDY OF THE PREDICTOR ALGORITHM
C *
C *
COMMON NORDER,ICONT,IRUN,IMAX,IPRINT
COMMON TM,DELM,TAU(10),DEL,TS,H,X(3000),TIME(3000)
DIMENSION P(3),STEP(3)
DIMENSION BUFFER(5000)
CALL PLOTS(BUFFER,5000)
1  READ1000,IPL0T,ICONT,IRUN,ISERCH,TS,TM,DELM,P(1),P(2),TMAX
  IF(IPL0T.EQ.9)CALL PLOT(0.0,0.0,0.999)
  IF(IPL0T.EQ.9)G0T0999
  READ1005,NORDER,DEL,(TAU(I),I=1,NORDER)
  PRINT1001
1001 FORMAT(1HO)
  PRINT1000,IPL0T,ICONT,IRUN,ISERCH,TS,TM,DELM,P(1),P(2)
1000 FORMAT(1X,4I1,5X,6F10.3)
  PRINT1005,NORDER,DEL,(TAU(I),I=1,NORDER)
1005 FORMAT(1X4,14F5.1)
  H=.01
  IMAX=TMAX/H+.5
  STEP(1)=.1
  STEP(2)=.1
  STEP(3)=.1
  XMAX=P(1)+1.0
  XMIN=0.0
IPRINT=0
IF(ISERCH.EQ.1)CALL GOLDN(XMAX,XMIN,100,P(1),CFUN)
IF(ISERCH.EQ.2)CALL PATTERN(2,P,STEP,2,2,CFUN)
IPRINT=1
CALL PROC(P,CFUN)
DO10 I=1,IMAX
IF(ABS(X(I)).GT.5.)X(I)=X(I)/ABS(X(I))*5.0
10 CONTINUE
X(IMAX+1)=0.0
X(IMAX+2)=0.5
TIME(IMAX+1)=0.0
TIME(IMAX+2)=IMAX/6.0
IF(IPLLOT.EQ.0)GOTO1
IF(IPLLOT.EQ.1)GOTO20
IF(IPLLOT.EQ.2)CALL PLOT(8.0,0.0,-3)
IF(IPLLOT.EQ.3)CALL PLOT(0.0,-7,-3)
IF(IPLLOT.EQ.4)CALL PLOT(8.0,7.0,-3)
CALL AXIS(0.0,-1.,8HRESPONSE,8,4.0,90.,-5.,5.,10.)
CALL AXIS(0.0,-1.,4HTIME,-4,6.0,0.0,0.0,TMAX/6.0,10.)
CALL PLOT(0.0,0.0,-3)
20 CALL LINE(TIME,X,IMAX,1,0,1)
CALL PLOT(0.0,0.0,-3)
GOTO1
999 STOP
END
PROGRAM VII-2

SUBROUTINE PROC(P,IAE)

C * * * * * * * * * * * * * * * * * * * * * * * * * * * * 
C * 
C * PREDICTOR ALGORITHM - GENERAL PROCESS LOOP 
C * 
C * 
C * THIS SUBROUTINE CONTAINS THE SIMULATION OF A GENERAL PROCESS LOOP 
C * CONTAINING THE PREDICTOR ALGORITHM AS WELL AS THE CONVENTIONAL P AND 
C * PI ALGORITHMS FOR THE PURPOSE OF EVALUATION AND COMPARISON. 
C * 
C * 
C * 
C------PROGRAM TO EVALUATE 1ST ORDER PREDICTOR ALGORITHMS AND COMPARE WIT 
C------CONVENTIONAL PROPORTIONAL AND PROPORTIONAL * INTEGRAL CONTROLLERS 
C

COMMON ICONT,IRUN,IMAX,TP1,TP2,TM,DEL,TS,H,X(2000),TIME(2000) 
COMMON IPLANT,NORDER,DEL2,TAU(11) 
DIMENSION P(3),NORDER,DEL2,TAU(11) 
DIMENSION XX(11) 
REAL IAE 
NPS=TS/H+.5 
PS=NPS 
ND=DEL/H+.5 
ND2=DEL2/H+.5 
N2=NORDER+1 
IF(IPLANT.EQ.2)ND=ND2 
IF(ND.EQ.0)ND=1 
HTS=TS 
IF(TS.LE.H)HTS=H 
C------CONSTANTS NECESSARY FOR PREDICTOR CONTROLLERS
N=DEL/TS + .001
XN=N
NXN=(DEL/TS-XN)*PS+.5
XN=NXN
XN=XN/PS
A=EXP(-TS/(2.*TM))
B=EXP(-TS/TM)
C=EXP(-TS/TM*XN)
SB=0
NN=N
IF(N.EQ.0)NN=1
DO1 L=1,NN
1    SB=SB+B***(L-1)
    IF(N.EQ.0)SB=0.
    BN=B**N
    P1=A/(1.-A)
    IF(ICONT.EQ.9)P1=P(1)
    IF(ICONT.EQ.9)ICONT=1
    GAIN1=P1/(1.+P1*(1.-A))
C------INITIAL CONDITION
SPT=0.0
GOTO(2,3),IRUN
2    DISTRB=0.0
    IF(ICONT.LE.3)SPO=(P1+1.)/P1
    IF(ICONT.EQ.4)SPO=(P(1)+1.)/P(1)
    IF(ICONT.EQ.5)SPO=1.0
    GOTO4
3    SPO=0
    DISTRB=1.0
4    SUM=0.
    T=0.0
    J=0
    I=0
    XG=0.0
X00 = 0.
XN = 0.
XT = 0.
XN1 = 0.
XBN = 0.
XBN1 = 0.
DO = 0
DT = 0.
U0 = 0.
UT = 0.
DO5 L = 1, 100
D(L) = 0.
5 U(L) = 0.
IAE = 0.
DO6 L = 1, ND
6 X(L) = 0.
DO7L = 1, 11
7 XX(') = 0.
GM = 0.
ITS = -1
C------PROCESS LOOP
100 ITS = ITS + 1
XTS = ITS
DO200 II = 1, NPS
I = I + 1
IF(I.GT.IMAX)GOTO999
XII = II
T = XII * H
TIME(I) = XTS * TS + T
IF(IPLANT.GT.1)GOTO170
150 IND = I + ND
C------FIRST ORDER PROCESS
IF(IND.LE.1)IND = I
X(IND) = (1 - EXP(-T)) * (U0 + DISTRB) + X00 * EXP(-T)
GO TO 200

C------HIGH ORDER PROCESS

170   XX(1)=U0+DISTRB

DO171  L=2,N2

171   XX(L)=XX(L-1)+(XX(L-1)-XX(L))*H/TAU(L-1)

IND=IND+1

X(N)=XX(N)

CONTINUE

X0=X(N)

IF(J.GT.N+2)J=1

IF(J.GT.N+2) J=1

K1=J+1

K2=J

IF(K1.GT.N+2)K1=K1-N-2

IF(K2.GT.N+2)K2=K2-N-2

IF(N.EQ.2)GO TO 290

IAE=IAE+ABS(1.-XO)

GOTO 290

IF(I .EQ. 1 OR I .EQ. 4)GOTO 294

IAE=IAE+ABS(XO)

GOTO 295

IF(I.AE.LT.1.0E20)GOTO 300

IAE=IAE*(1.+1./TIME(I))

GOTO(10,20,30,40,50),I CONT

C------PROPORTIONAL PREDICTOR--LOAD IGNORED

10   SBU=0.0

DO15  L=1,NN

K=J-L

IF(K.LT.1)K=N+2+K

15   SBU=SBU+B**((L-1)*U(K)

IF(N.EQ.0)SBU=0

XBN=(1.-B)*SBU+BN*(1.-C)*U(K1)+C*XO
PARAMETERS

G0 = GAIN1

C------PROPORTIONAL PREDICTOR--LOAD CALCULATED

20 XPO = -(B * X + B * (1 - 1 / C) * (U(K2) + D0) - (1 - B / C) * (U(K1) + D0))

SUM = SUM + (X0 - XPO) * P(1) * HTS / (1 - B)

DO = SUM

PRINT69, X0, XPO, DO

SBU = 0

DO25L = 1, NN

K = j - L

IF (K .LT. 1) K = N + 2 + K

25 SBUD = SBUD + B * (L - 1) * (U(K))

IF (N .EQ. 0) SBUD = 0.0

XBN = (1 - B) * (SB * D0 + SBUD) + BN * ((1 - C) * (U(K1) + D0) + C * X0)

UO = GAIN1 * (SPO - A * XBN) - DO

GOTO199

C------PROPORTIONAL PREDICTOR--LOAD MEASURED

30 SBUD = 0.0

DO35L = 1, NN

K = J - L

IF (K .LT. 1) K = N + 2 + K

35 SBUD = SBUD + B * (L - 1) * (U(K) + D(K))

IF (N .EQ. 0) SBUD = 0.0

XBN = (1 - B) * SBUD + BN * ((1 - C) * (U(K1) + D(K1)) + C * X0)

UO = GAIN1 * (SPO - A * XBN) - DISTRB

GOTO199

40 UO = P(1) * (SPO - XO)

GOTO199

C------PROPORTIONAL + INTEGRAL CONVENTIONAL

50 SUM = SUM + (SPO - XO) * HTS

UO = P(1) * (SPO - XO) + P(2) * SUM

199 U(I,J) = UO

D(J) = DISTRB

69 FORMAT(1X6F15.5)
UT=U0
XT=X0
SPT=SPO
DT=D0
GOTO100
999 CONTINUE
68 FORMAT(1X,E20.5,2F15.5)
RETURN
END
PROGRAM VII-4

SUBROUTINE GOLDN(XM,XN,FT,X,CFUN)

C * PREDICTOR ALGORITHM - ONE-DIMENSIONAL SEARCH *

C * THIS SUBROUTINE CONTAINS THE GOLDEN SECTION ONE-DIMENSIONAL SEARCH *
C * LOGIC USED TO TUNE THE PROPORTIONAL CONTROLLER. *

C

XMAX=XM
XMIN=XN
FINT=FT
DIMENSION ER(2),P(2)
DIMENSION PP(3)

C----PLACEMENT OF TWO INITIAL STARTING POINTS
P(1)=XMIN+.618*(XMAX-XMIN)
PP(3)=0.0
PP(2)=0.0
PP(1)=P(1)
CALL PROC(PP,ER(1))
P1=P(1)
ER1=ER(1)
GOTO55

C----SEQUENTIAL SEARCH LOOP
13 IF(ER1-ER(2))15,15,14
14 ER1=ER(2)
P1=P(1)

C----MAKING SURE P(2) IS LARGER THAN P(1)
15 IF(P(2)-P(1))17,20,20
C----DECIDING WHERE TO PLACE NEXT EXPERIMENT
17 PD=P(1)
   ERD=ER(1)
   ER(1)=ER(2)
   ER(2)=ERD
   P(1)=P(2)
   P(2)=PD

20 IF(ER(1)-ER(2))40,40,30

30 XMIN=P(1)
   P(1)=P(2)
   ER(1)=ER(2)
   GOT050

40 XMAX=P(2)

50 IF(XMAX-XMIN.LE.FINT)GOT090

55 PXX={(P(1)-XMIN)/(XMAX-XMIN)}
   IF(PXX.GE.50)P(2)=XMIN+.382*(XMAX-XMIN)
   IF(PXX.LT.0.5)P(2)=XMIN+.618*(XMAX-XMIN)

80 PP(1)=P(2)
   CALL PROC(PP,ER(2))
   GOT013

90 CFUN=ER1
   X=P1
   RETURN
END
Vita

Charles F. Moore is the son of Mr. and Mrs. Fred Moore of Plaquemine, Louisiana. He was born in Morrilton, Arkansas on June 18, 1943. He graduated from Plaquemine High School in 1961.

He attended Louisiana State University on a football athletic scholarship and graduated in May, 1966 with a Bachelor of Science Degree in Chemical Engineering. He received his Master of Science Degree in Chemical Engineering from Louisiana State University in January, 1968.

In December, 1963 he married the former Kitty Breaux of Plaquemine, Louisiana. He is the father of two children, a son Rhett, born June 6, 1966 and a daughter Daley, born May 20, 1968.

He is now a candidate for the degree of Doctor of Philosophy in the field of Chemical Engineering.
Candidate: Charles Fred Moore

Major Field: Chemical Engineering

Title of Thesis: Selected Problems in the Design and Implementation of Direct Digital Control

Approved:

Paul W. Merrill
Major Professor and Chairman

R.D. Anderson
Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination:

December 18, 1968