Stochastic Control of Chemical Processes.

Brian Lee Ramaker

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/1514
This dissertation has been microfilmed exactly as received 69-4497

RAMAKER, Brian Lee, 1941-
STOCHASTIC CONTROL OF CHEMICAL PROCESSES.

Louisiana State University and Agricultural and Mechanical College, Ph.D., 1968
Engineering, chemical

University Microfilms, Inc., Ann Arbor, Michigan
STOCHASTIC CONTROL OF CHEMICAL PROCESSES

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

Brian Lee Ramaker
B.S., University of Wisconsin, 1963
M.S., University of Wisconsin, 1964
August, 1968
ACKNOWLEDGEMENT

This research was conducted under the guidance of Dr. Cecil L. Smith, Jr., Associate Professor of Chemical Engineering at Louisiana State University. The author is highly appreciative for the consideration, encouragement, and interest shown throughout this project. The author is very proud to be his first doctoral candidate.

Thanks are also extended to other members of the Chemical Engineering Department for their assistance, as well as to Dr. J. R. Johnson and Dr. J. L. Hilburn of the Electrical Engineering Department. Much appreciation is extended to the personnel at 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 NASA for the traineeship which supported the author for the last two years.

Sincere thanks are also extended to Miss Madeline Jewell for her assistance with the preparation of the 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.
TABLE OF CONTENTS

ACKNOWLEDGEMENT ........................................... ii
LIST OF TABLES ............................................. vi
LIST OF FIGURES ............................................ viii
ABSTRACT .................................................... x

CHAPTER

I  INTRODUCTION ........................................ 1
  Literature Cited ....................................... 4

II DETERMINATION OF DYNAMIC MODEL
  PARAMETERS USING QUASILINEARIZATION ............. 5
  Programming Consideration ......................... 13
  Identification of the Dead Time ................... 16
  Effect of Data on Convergence ..................... 21
  Minimizing Number of Calculations ............... 23
  Increasing the Parameter Space of Convergence 27
  Reduction of Step Size .............................. 31
  Summary ............................................... 32
  Literature Cited ..................................... 35

III THEORETICAL DEVELOPMENT OF PREDICTOR
  MODELS ............................................... 37
  Stochastic Model .................................... 37
  Dynamic Model ....................................... 39
  Determination of Parameters ..................... 42
<table>
<thead>
<tr>
<th>Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fletcher Powell</td>
<td>221</td>
</tr>
<tr>
<td>Marquardt's Method</td>
<td>231</td>
</tr>
<tr>
<td>Pattern Search</td>
<td>243</td>
</tr>
<tr>
<td>Powell's Method</td>
<td>248</td>
</tr>
<tr>
<td>Powell's Method With Fibonacci Search</td>
<td>256</td>
</tr>
<tr>
<td>Rosenbrock's Method I</td>
<td>264</td>
</tr>
<tr>
<td>Rosenbrock's Method II</td>
<td>268</td>
</tr>
<tr>
<td>D COMPUTER PROGRAM FOR PATTERN SEARCH ON</td>
<td>273</td>
</tr>
<tr>
<td>SIX PARAMETER MODEL</td>
<td></td>
</tr>
<tr>
<td>E COMPUTER PROGRAM FOR PREDICTIVE CONTROL</td>
<td>286</td>
</tr>
<tr>
<td>OF A SECOND ORDER SYSTEM</td>
<td></td>
</tr>
<tr>
<td>F COMPUTER PROGRAM FOR PREDICTIVE CONTROL</td>
<td>292</td>
</tr>
<tr>
<td>OF A CHEMICAL REACTOR</td>
<td></td>
</tr>
<tr>
<td>VITA</td>
<td>298</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Effect of Order of Model on Convergence</td>
<td>19</td>
</tr>
<tr>
<td>2.2</td>
<td>Effect of Data on Convergence</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>Effect of Program Changes on Convergence</td>
<td>26</td>
</tr>
<tr>
<td>2.4</td>
<td>Comparison of Convergence Using Marquardt's Method</td>
<td>30</td>
</tr>
<tr>
<td>2.5</td>
<td>Comparison of Convergence Using Reduced Step Size</td>
<td>33</td>
</tr>
<tr>
<td>4.1</td>
<td>Four Parameter Search</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>Three Parameter Search</td>
<td>55</td>
</tr>
<tr>
<td>4.3</td>
<td>Six Parameter Search</td>
<td>58</td>
</tr>
<tr>
<td>4.4</td>
<td>Effect of Forcing Function on $\delta_1$ and $\delta_2$</td>
<td>67</td>
</tr>
<tr>
<td>4.5</td>
<td>Effect of Number of Data Points on $\delta_1$ and $\delta_2$</td>
<td>68</td>
</tr>
<tr>
<td>4.6</td>
<td>Comparison of System Time Constants With $\delta_1$ and $\delta_2$</td>
<td>71</td>
</tr>
<tr>
<td>5.1</td>
<td>Optimally Tuned Predictor Controller</td>
<td>91</td>
</tr>
<tr>
<td>5.2</td>
<td>Control of Second Order System With Dead Time</td>
<td>93</td>
</tr>
<tr>
<td>5.3</td>
<td>The Effect of Less Frequent Sampling</td>
<td>96</td>
</tr>
<tr>
<td>5.4</td>
<td>Comparison of Predictor Controller and PI Controller</td>
<td>98</td>
</tr>
<tr>
<td>5.5</td>
<td>Noise on Reactor Feeds</td>
<td>105</td>
</tr>
<tr>
<td>5.6</td>
<td>Determination of Chemical Reactor Parameters</td>
<td>108</td>
</tr>
<tr>
<td>5.7</td>
<td>Predictive Control of Chemical Reactor</td>
<td>110</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.8</td>
<td>Comparison of PI and Predictor Control Of a Chemical Reactor</td>
<td>114</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>General Program Outline of Quasilinearization</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>Dead Time M(p-6)</td>
<td>18</td>
</tr>
<tr>
<td>2.3</td>
<td>Solution-Assume Program Outline of Quasilinearization</td>
<td>24</td>
</tr>
<tr>
<td>3.1</td>
<td>Description of Computer Control System</td>
<td>38</td>
</tr>
<tr>
<td>3.2</td>
<td>Dead Time M(p-λ)</td>
<td>41</td>
</tr>
<tr>
<td>4.1</td>
<td>Two Dimensional Contour δ₂ vs δ₁ (Noise RMS = .1)</td>
<td>61</td>
</tr>
<tr>
<td>4.2</td>
<td>Two Dimensional Contour λ vs g (Noise RMS = .1)</td>
<td>62</td>
</tr>
<tr>
<td>4.3</td>
<td>Two Dimensional Contour δ₂ vs δ₁ (Noise RMS = .2)</td>
<td>63</td>
</tr>
<tr>
<td>4.4</td>
<td>Two Dimensional Contour λ vs g (Noise RMS = .2)</td>
<td>64</td>
</tr>
<tr>
<td>4.5</td>
<td>Number of Necessary Data Points vs RMS Value of Noise</td>
<td>70</td>
</tr>
<tr>
<td>4.6</td>
<td>Model Error on Experimental Data (Noise RMS = .2)</td>
<td>76</td>
</tr>
<tr>
<td>4.7</td>
<td>Model Error on Experimental Data (Noise RMS = .5)</td>
<td>77</td>
</tr>
<tr>
<td>4.8</td>
<td>Model Error On Experimental Data (Noise RMS = 1.)</td>
<td>78</td>
</tr>
<tr>
<td>5.1</td>
<td>Sensitivity of Control to Model Parameters</td>
<td>85</td>
</tr>
<tr>
<td>5.2</td>
<td>Effect of Model Parameter Error on Manipulated Variable</td>
<td>86</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.3</td>
<td>Sensitivity of Control to Controller Parameters</td>
<td>88</td>
</tr>
<tr>
<td>5.4</td>
<td>Effect of Control Parameter Error on Manipulated Variable</td>
<td>89</td>
</tr>
<tr>
<td>5.5</td>
<td>Change in Manipulated Variable as the Dead Time Increases</td>
<td>94</td>
</tr>
<tr>
<td>5.6</td>
<td>Predictive Control of Second Order System (Noise RMS = .2)</td>
<td>99</td>
</tr>
<tr>
<td>5.7</td>
<td>Predictive Control of Second Order System For a Step Change in Disturbance of 10 (Noise RMS = .2)</td>
<td>100</td>
</tr>
<tr>
<td>5.8</td>
<td>Predictive Control of Second Order System For a Step Change in Set Point of 1 (Noise RMS = .2)</td>
<td>101</td>
</tr>
<tr>
<td>5.9</td>
<td>Predictive Control of Chemical Reactor</td>
<td>102</td>
</tr>
<tr>
<td>5.10</td>
<td>Feed Stream Fluctuations to the Chemical Reactor</td>
<td>106</td>
</tr>
<tr>
<td>5.11</td>
<td>Comparison of Predictive Control and PI Control of a Chemical Reactor</td>
<td>113</td>
</tr>
<tr>
<td>A-3</td>
<td>Gradient Partan</td>
<td>121</td>
</tr>
</tbody>
</table>
The work in this dissertation is divided into two areas: (1) the
determination of models that adequately describe the system for control
purposes from experimental input-output data and (2) the use of these
models in a direct digital control scheme to control the system. The
entire study was done on an IBM 7040 computer including the generation
of the experimental data.

One of the methods used to identify the deterministic portion of
the system was quasilinearization. Starting with an initial trial
solution to the differential equations the problem is formulated as
a sequence of linear problems whose solution converge to the non-linear
solution. This method correctly identified the parameters in a dy-
namic model including dead time up to third order from input-output
data. The order of the equation for which quasilinearization will be
successful is limited by the fact that only one state variable is
observed and the others must be calculated. The number of calculations
and thus the computer running time for this particular problem was
greatly reduced by defining the state variables in terms of the change
in the parameters and by using the model equation instead of the funda-
mental solutions to update the output and its derivatives. The para-
meter space in which convergence will occur for this particular prob-
lem was demonstrated to be increased by the incorporation of Marquardt's
method into quasilinearization or by using only a portion of the step
size calculated by quasilinearization.

Recent work by Box and Jenkins suggest a form of a deterministic model for the system dynamics and a stochastic model for the disturbance. Using these models an extensive comparison of optimization techniques including: Optimum Gradient, Gradient Partan, Davidon, Fletcher Powell, Marquardt, Rotational Discrimination, Pattern, Powell, and Rosenbrock was made as applied to the specific problem of determining the model parameters. It was found that Pattern Search was the best of the methods tried due to its reliability, simplicity and efficiency in the limited parameter space.

The control strategy used in the second portion of this investigation was that proposed by Box and Jenkins. The stochastic model was used to predict the error from set point some number of sampling periods in the future. The dynamic model was used to calculate the change in the manipulated variable which will prevent the system from being off target by this predicted amount. Applying this control strategy to second order systems with dead times of greater than .5 sampling periods, control was unstable. Decreasing the sampling rate was found to be the best solution. The predictor controller showed on the average a 50% improvement over an optimally tuned PI controller for a noisy system and step changes in both disturbances and set point.

The complete procedure for obtaining the model and controlling the system was demonstrated on a chemical reactor. It was found that although a good fit of the experimental data could be obtained that equivalent control was not always possible due to the limited amount of change of the manipulated variable.

xi
CHAPTER I

INTRODUCTION

A system needs to be controlled because it is subject to disturbances which force it away from the best or optimal operating conditions. This requirement has given rise to conventional analog controllers and, more recently, to digital computers for process control. Possession of a mathematical model that adequately describes the physical or chemical system or process will facilitate implementation of control and optimization of the system. Such models must accurately describe complicated systems, yet be simple enough to permit rapid calculations. Computers have made it possible to perform calculations rapidly and have, therefore, become important control devices.

With all the potentials of the computer available, it would be desirable if the computer could (1) identify the system (including both deterministic and stochastic characteristics), (2) compute the control strategy, and (3) implement the control. This may sound too idealistic but there are possibilities.

The method of quasilinearization as proposed by Bellman and Kalaba (1) can be used to identify the deterministic portion of the system. The parameters estimation is treated as a multipoint boundary value problem. Starting with an initial trial solution to the differential equations the problem is formulated as a sequence of linear
problems whose solutions converge to the non-linear solution.

Recent work by Box and Jenkins (2, 3, 4, 5, 6) describes a method that potentially would make it possible for the computer to perform the three desired operations. The computer ascertains the deterministic and stochastic characteristics of the system under control from the normal input to the computer (the feedback variable) and output from the computer (the manipulated variable). The computer then incorporates the parameters for this derived mathematical model into a control equation.

Chapter II deals with the development of quasilinearization as it applies to the specific problem of determining the parameters of a dynamic model from process input-output data. A computational procedure was developed where by systems up to third order including dead time could be correctly identified. The inherent problems of this method including large storage requirements, large number of computations and limited parameter space of convergence were investigated. A number of ideas were tested which seem to alleviate some of these problems for the cases tested.

Chapter III reviews the background for the method proposed by Box and Jenkins. General models are developed for both the dynamic response of the system and the stochastic nature of the disturbance. Chapter IV discusses specific computational aspects involved in determining the model parameters. Inherent in the results is an extensive comparison of optimization techniques. Thus, this work has entailed trials of the various optimization techniques available in the literature for this specific problem. The objective was not to determine
a good optimization technique in general, but one that works well for this specific problem.

Chapter V uses the models developed in the previous chapters to design an optimal control scheme. The stochastic model is used to predict in advance what the next disturbance will be. The dynamic model is then used to calculate the change in the manipulated variable which will prevent this predicted deviation from occurring. This control strategy is used to investigate the control problems of a second-order system including dead time and a non-linear chemical reactor. A comparison is made between this predictive scheme and an optimally tuned proportional plus integral controller for these two noisy systems.

In summarizing, the research work discussed in this dissertation can be divided into two areas: (1) determination of models that adequately describe the system from experimental input-output data and (2) the use of these models in a direct digital control scheme to control the system.
LITERATURE CITED


CHAPTER II
DETERMINATION OF DYNAMIC MODEL
PARAMETERS USING QUASILINEARIZATION

System identification, or the development of a model for a given system, has become quite important in all fields of engineering. The possession of an adequate model facilitates implementation of control and optimization of the system, as well as giving insight into its operation. The forms of the differential equations describing various systems are fairly well documented; however, the parameters must often be obtained from experimental data. These parameters generally cannot be measured directly but must be determined from input-output data taken at intervals of time. The problem now becomes one of finding the parameters which give the best fit to discrete experimental data for a set of linear or nonlinear differential equations. The method of system identification called quasilinearization, developed by Bellman and Kalaba (1), treats this problem as a multipoint boundary value problem. Starting with an initial trial solution, the problem is formulated as a sequence of linear problems whose solutions converge to the non-linear solution.

This chapter discusses the application of quasilinearization to the specific problem of determining the parameters in a dynamic model. The model includes parameters related to the system time constants, the gain, and the dead time (transportation lag or time delay). The
objectives of this research are:

1. To show quasilinearization could be used to determine all model parameters including the dead time.

2. To determine the effect of data length and the number of significant figures on the convergence.

3. To minimize the number of calculations per iteration.
   a. Calculate the solution in terms of parameter change rather than the parameters themselves, which removes the necessity of a particular solution.
   b. Use the model equation to update the output and its derivatives instead of the fundamental solutions.

4. To investigate various methods to increase the parameter space in which convergence will occur including:
   a. Incorporation of Marquardt's Method into quasilinearization.
   b. Taking only a portion of the calculated step.

To illustrate the approach of quasilinearization and to establish the notation that will be used in this chapter, consider the $N^{th}$ order linear differential equation generally used for control purposes to describe system dynamics (including dead time):

$$a_N \frac{d^N x(t)}{dt^N} + \ldots + a_1 \frac{dx(t)}{dt} + x(t) = gM(t-\theta)$$  \hspace{1cm} 2.1

The $a$'s are related to the time constants of the process, $g$ is the process gain, and $\theta$ is the dead time. This equation can be written in state variable form by choosing the $N$ state variables to be $x(t)$ and its derivatives:
\[ \ddot{\mathbf{x}}(t) = A \dot{\mathbf{x}}(t) + B(t) \]
\[ Y(t) = C \mathbf{x}(t) \]

where:

\[ \ddot{\mathbf{x}}(t) = \begin{bmatrix} x(t) \\ \frac{dx(t)}{dt} \\ \frac{d^2x(t)}{dt^2} \\ \vdots \\ \frac{d^{N-1}x(t)}{dt^{N-1}} \end{bmatrix} \]

\[ A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ - \frac{1}{a_N} & - \frac{a_1}{a_N} & - \frac{a_2}{a_N} & \cdots & - \frac{a_{N-1}}{a_N} \end{bmatrix} \]

\[ B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \frac{g}{a_N} M(t-\theta) \end{bmatrix} \]
It is desired to determine the parameters \( a_1 \) thru \( a_N \), \( g \) and the dead time \( \theta \) from discrete observations on only the first state variable \( x(t) \) and the manipulated variable \( M(t) \).

As the initial conditions for all of the state variables are not known, but only a record of discrete input-output data, the parameter estimation must be formulated as a multipoint boundary value problem. The mathematical treatment and solution of linear boundary value problems is well understood. It is therefore desirable to formulate nonlinear boundary value problems in such a way that linear techniques of solution may be used. Quasilinearization, which stems from Kantorovich's extension of the well-known Newton-Raphson procedure to function space, treats the non-linear problem as a limit of a sequence of linear problems (2).

Since in the original problem the parameters \( a_1, a_2, \ldots, a_N, g \) and \( \theta \) are unknowns they can be treated as additional state variables. As these state variables are constant with respect to time, they are described by the differential equations:

\[
\frac{da_1}{dt} = \frac{da_2}{dt} = \ldots = \frac{da_N}{dt} = 0, \quad \frac{dg}{dt} = 0, \quad \frac{d\theta}{dt} = 0
\]

However, Equation 2.1 is now no longer linear with respect to the state variables. To illustrate the method used by quasilinearization to linearize non-linear differential equations consider a first order system described by the equation:

\[
\frac{dx(t)}{dt} = -\frac{1}{a_1} x(t) + \frac{g}{a_1} M(t-\theta) = -cx(t) + GN(t-\theta)
\]
where
\[ c = \frac{1}{a} \quad \text{and} \quad G = \frac{c}{a} \]

The constants \( c \), \( G \) and the dead time \( \theta \) are to be determined from discrete observations on \( x(t) \) and \( M(t) \). As the unknown parameters are treated as state variables, the number of state variables is increased from \( N \) to \( m \) where \( m = 2N+2 \). Application of the generalized Newton-Raphson Formula:

\[
 f(x_{n+1}) = f(x_n) + f'(x_n) (x_{n+1} - x_n)
\]

to Equation 2.4 yields:

\[
 x'_{n+1} = -c_n x_n + G M(t-\theta_n) +
\]

\[
 (x_{n+1} - x_n) (-c_n) + (c_{n+1} - c_n) (-x_n) +
\]

\[
 (G_{n+1} - G_n) M(t-\theta_n) + (\theta_{n+1} - \theta_n) G_n [-M'(t-\theta_n)]
\]

The remaining differential equations which describe this problem are:

\[
 \frac{dc_{n+1}}{dt} = \frac{dG_{n+1}}{dt} = \frac{d\theta_{n+1}}{dt} = 0
\]

and are already linear.

By defining the \( m \) state variables to be the terms in the \( n+1 \)st

solution

\[
 \begin{bmatrix}
 x_{n+1} \\
 c_{n+1} \\
 G_{n+1} \\
 \theta_{n+1}
\end{bmatrix}
\]

the equations are represented by the linear equation

\[
 \dot{x}_{n+1} = A \dot{x}_n + B x
\]
and the A matrix and B vector become:

\[
A_x = \begin{bmatrix}
-c_n & -x_n & M(t-\theta_n) & -G_n M'(t-\theta_n) \\
0 & 0 & 0 & 0 \\
0 & -c & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
B_x = \begin{bmatrix}
c_n x_n + \theta_n M'(t-\theta_n) \\
0 \\
0 \\
0
\end{bmatrix}
\]

If the \( n \) state variables are defined in terms of the change of the state variables from one solution to the next, the equations become homogeneous, making it unnecessary to calculate a particular solution. The necessary equations are:

\[
\bar{\delta}' = A_\delta \bar{\delta}
\]

\[
\bar{x}_{n+1} = \bar{x}_n + \bar{\delta}
\]

\[
\begin{bmatrix}
x_{n+1} - x_n \\
c_{n+1} - c_n \\
G_{n+1} - G_n \\
\theta_{n+1} - \theta_n
\end{bmatrix}
\]
Formulation of the problem in terms of the changes of the parameters is thus very desirable and significantly reduces the number of calculations that must be made.

During each iteration of quasilinearization it will be necessary to numerically determine the solution to this set of linear first order differential equations. Appearing in this set of equations is the derivative of the manipulated variable, which is not directly available. However, the integrated form of the solution will not contain this derivative, thus making it possible for this method to determine the system dead time.

Consider the differential equation

\[ \dot{x} = A x + B \]

where the order of the matrixes and vectors is m. The homogeneous solution can be expressed as a linear combination of the fundamental solution \( \mathbf{Y}_{h_i} \) obtained by solving the equation

\[ \dot{\mathbf{Y}}_{h_i} = A \mathbf{Y}_{h_i} \]  \hspace{1cm} 2.7

with the m sets of initial conditions

\[
A_{\delta} = \begin{bmatrix}
-c_n & -x_n & M(t-\theta_n) & -G_n M'(t-\delta_n) \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 
\end{bmatrix}
\]
The particular solution $\overline{Y}_p$ is the solution of the equation
\[ \overline{Y}_p' = A\overline{Y}_p + B \] with the initial condition
\[ \overline{Y}_p(0) = 0 \]

The solution for $\overline{x}_{n+1}$ now becomes
\[ \overline{x}_{n+1} = \sum_{j=1}^{m} \mu_i \overline{Y}_h_i + Y_p \] where the constants $\mu_i$ are determined from the boundary conditions on $x$.

In the above discussion it was shown that quasilinearization could be applied by either of the following equations:
\[ \overline{x}_{n+1}' = A_x \overline{x}_{n+1} + B_x \]
or
\[
\delta' = A \delta
\]
\[
\bar{x}_{n+1} = \bar{x}_n + \delta
\]

Obviously using the second formulation eliminates the need for calculating the particular solution, and thus reduces the calculations.

The coefficients \( \mu_i \) in the linear combination of the fundamental set of solutions must be determined from the available experimental observations. Quasilinearization treats this as an overspecified multipoint boundary value problem. Thus the coefficients are chosen to minimize the sum of squares of the deviations from the observed points, or mathematically,

\[
\min \sum_{i=1}^{k} [x_{n+1}(t_i) - x_{obs}(t_i)]^2
\]

This can be applied with equal facility to either of the above formulations.

Programming Considerations

The computer program to determine the parameters in the dynamic model using the method of quasilinearization was written in Fortran IV for use on a 32K IBM 7040. Up to sixth order differential equations of the general form in Equation 2.1 could be used for the model.

One of the inherent problems with this method is the large demands placed on core storage for storing the fundamental solutions (Equation 2.7). This can be circumvented by sacrificing computer running time to reduce core storage requirements, which was done in the program. For example, if the model is third order including the dead
time and 200 data points are used with five points being calculated between each data point to improve the accuracy of the integration, the storage of the fundamental solution would require 72K. The program whose outline is shown in Figure 2.1 requires only 3072 storage locations for the same purpose but the fundamental solution must be generated twice. Each of these blocks perform a basic operation required by quasilinearization and are written in subroutine form.

To initialize the iterations for quasilinearization requires assumptions of values for each state variable at every point in time corresponding to observed data points. The function of the subroutine ASSUME is to generate this initial solution for any given set of parameters and forcing function. The missing state variables, the output, and the necessary derivatives are calculated from the differential equation being used as the model. These values of the state variables must be stored for later use.

The most time-consuming calculation during each iteration of quasilinearization is the numerical calculation of the fundamental solutions. The function of the subroutine SOLUTION is to generate these fundamental solutions. To minimize the storage requirement all the homogeneous solutions and the particular solution, if used, are generated simultaneously with only the most recent value being retained. The calculations are kept to a minimum by taking advantage of the many zeroes that appear in the matrices.

Since the complete solutions are not stored, it is necessary to calculate the fundamental solutions twice during each iteration. The first time the solutions are calculated the necessary information
Figure 2.1. General Program Outline of Quasilinearization.
is generated to calculate the new parameter values. The subroutine
MATRIX takes this information and forms m algebraic equations which
are solved simultaneously in subroutine GAUSS by Gauss reduction
(using row interchange) for the new parameters. The second time the
solution is generated the new parameters are used and subroutine
REGEN is used to update the remaining state variables, namely the out­
put and its derivatives. By generating the solutions twice consider­
able computer storage is saved but the calculation time is nearly
doubled. However, this was faster than using bulk storage. The con­
vergence is then checked to see if the parameters changed sufficiently
during the last iteration to warrant going through the iterative
process again.

All numerical integration throughout this program was done using
rectangular integration. To increase the accuracy, the interval be­
tween observations was subdivided. The forcing function used in each
of these subintervals was the linearly interpolated value between
the observed values. Melsa, Pillmeir, Bottorff, and Steinway (6)
report a survey of numerical integration methods on quasilineariza­
tion including: Newtonian, Gaussian quadrature, backward differences,
and rectangular integration. Their conclusion was that rectangular
integration is adequate.

Identification of the Dead Time

The term M(t-θ) which appears in the differential equations is
 calculated by looking back in time a period of θ sampling periods.
If the dead time is not an integral number of sampling periods linear
interpolation is used between the two nearest points. M is therefore
replaced by \((q+1-6)M_{p-q} + (\theta-q)M_{p-q-1}\) as illustrated in Figure 2.2, where \(q\) is the integral number of sampling periods in \(\theta\). It is possible that on certain iterations as the solution is approached that a negative dead time will appear. Although this situation is physically unrealizable for real systems, this does not interfere with the convergence procedure.

The convergence of quasilinearization is quadratic and very rapid when it does occur. Unfortunately the parameter space, for this particular problem in which convergence will occur is rather narrow. Table 2.1 shows the effect on convergence of increasing the order of the model. The data for each test was generated from the appropriate order model equation using a damped sine as the forcing function. The correct parameters were therefore known and a perfect fit was possible. Table 2.1 shows that as the order of the model increases the parameter space in which convergence will occur becomes narrower. This means that the initial trial vector of parameters must be closer to the solution vector as the order of the model increases. Including dead time in the model, systems from first to third order were correctly identified. Fourth order data using 50 data points was impossible to identify even when the trial solution was extremely close to the true solution. The order of the equation for which quasilinearization will be successful is limited by the fact that only one state variable, namely \(x(t)\), is observed and the other \(m-1\) must be calculated. For this particular model equation including dead time, third order appears to be the limit. Table 2.1 also indicates that the dead time parameter \(\theta\) is more difficult to
Figure 2.2. Dead Time $M(p-0)$. 
### Table 2.1

**EFFECT OF ORDER OF MODEL ON CONVERGENCE**

<table>
<thead>
<tr>
<th>Order of Model</th>
<th>Order of Data</th>
<th>$c_4$</th>
<th>$c_3$</th>
<th>$c_2$</th>
<th>$c_1$</th>
<th>$G$</th>
<th>$G_0$</th>
<th>Converged</th>
<th>Iterations</th>
<th>Number of Data Points</th>
<th>Significant Fig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Parameters</td>
<td>1</td>
<td></td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>Starting Point</td>
<td>1</td>
<td>.6</td>
<td>.6</td>
<td>.6</td>
<td>No</td>
<td>50</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>.8</td>
<td>.8</td>
<td>.8</td>
<td>Yes</td>
<td>7</td>
<td>50</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>Yes</td>
<td>5</td>
<td>50</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correct Parameters</td>
<td>2</td>
<td></td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>Starting Point</td>
<td>2</td>
<td>.9</td>
<td>.9</td>
<td>.9</td>
<td>No</td>
<td>50</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>Yes</td>
<td>7</td>
<td>50</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correct Parameters</td>
<td>3</td>
<td></td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>Starting Point</td>
<td>3</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>Yes</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>Yes</td>
<td>5</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Yes</td>
<td>6</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Correct Parameters</th>
<th>3</th>
<th>2.5</th>
<th>2.5</th>
<th>2.5</th>
<th>2.5</th>
<th>2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting Point</td>
<td>3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.3</td>
<td>1.3</td>
<td>1.3</td>
<td>1.3</td>
<td>1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Correct Parameters</td>
<td>4</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>Starting Point</td>
<td>4</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>Correct Parameters</td>
<td>4</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>Starting Point</td>
<td>4</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
<td>2.4</td>
</tr>
</tbody>
</table>

**TABLE 2.1 CONTINUED**
determine than is an additional dynamic parameter. The third order system without dead time was correctly identified down to a starting point of zero. The same system with a dead time would not converge at a starting point of 1.0. On the other hand, in going from a second order system to a third order system very little difference was noted in the decrease of the parameter space. Also higher than third order systems, namely fourth, were correctly identified when the dead time was not included in the model.

**Effect of Data on Convergence**

The parameter space in which convergence will occur is affected very little by the number of data points used. Table 2.2 shows that for a third order system not including dead time that 20 data points were sufficient. A fourth order system including dead time could not be identified even when the number of data points was increased to 200. In the majority of tests 50 data points were used and if convergence did not occur increasing the amount of data did not help.

The accuracy of the final answer was a function of the number of properly spaced data points and the accuracy of the data. Table 2.2 shows the effect of reducing the number of significant figures. Convergence was not adversely affected by the number of significant figures as the same number of iterations were required whether 3, 2 or 1 significant figures were used. The accuracy of the final answer was however affected by the number of significant figures. By increasing the amount of data used the accuracy of the results was increased.
TABLE 2.2
EFFECT OF DATA ON CONVERGENCE

<table>
<thead>
<tr>
<th>Order of Model</th>
<th>Order of Data</th>
<th>c_4</th>
<th>c_3</th>
<th>c_2</th>
<th>c_1</th>
<th>G</th>
<th>0</th>
<th>Iteration</th>
<th>Converged</th>
<th>Number of Data Points</th>
<th>Significance Fig.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct Parameters</td>
<td>3</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>5</td>
<td>Yes</td>
<td>50</td>
<td>3</td>
</tr>
<tr>
<td>Starting Point</td>
<td>3</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>1.2</td>
<td>0</td>
<td>2.5</td>
<td>5</td>
<td>Yes</td>
<td>40</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>Yes</td>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
<td>Yes</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>No</td>
<td></td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Correct Parameters</td>
<td>3</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td>4</td>
<td>Yes</td>
<td>50</td>
<td>3</td>
</tr>
<tr>
<td>Starting Point</td>
<td>3</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>1.5</td>
<td>4</td>
<td>Yes</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>Yes</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>Yes</td>
<td>50</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>Yes</td>
<td>50</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>Yes</td>
<td>50</td>
<td>1</td>
</tr>
</tbody>
</table>
Minimizing Number of Calculations

The method of quasilinearization has thus been demonstrated to successfully determine the parameters in a dynamic model, up to third order including dead time, from input-output data. The convergence of this method, when it does occur, is quadratic and thus very rapid as far as the number of iterations is concerned. However, the number of calculations that must be performed during each iteration is extremely large. Because of the particular form of the differential equation for this problem and utilizing knowledge of the physical situation, the number of calculations can be reduced. Any reduction in the number of calculations will increase the efficiency of the method as far as computer running time is concerned.

Earlier in this chapter the deletion of the particular solution by defining the state variables in terms of the changes in the parameters was discussed. For a second order model not including dead time this reduces the number of calculations in the solution of the differential equations by 20%. Since about 90% of the computer running time is required to generate the fundamental solutions due to the large number of calculations and the amount of necessary indexing, a 20% reduction represents a significant decrease.

The necessity of having to generate all of the fundamental solutions the second time during each iteration to update the missing state variable can be circumvented by using subroutine ASSUME to regenerate the necessary output and its derivatives. (Figure 2.3) The number of calculations performed by ASSUME is equivalent to the generation of one of the fundamental solutions by subroutine SOLUTION.
Figure 2.3. Solution-Assume Program Outline of Quasilinearization.
For a second order system this means a savings of 35\% in computer running time. For higher order systems the savings will even be greater.

A second order system not including dead time was correctly identified by the original general program in five iterations requiring a total of 59 seconds computer running time. Taking into account all of the suggested reduction in calculations the computer running time per iteration for this particular problem could be reduced by 50\%. This means that within the parameter space where convergence will occur the method of quasilinearization would be competitive with other techniques for solving the same problem, such as optimum seeking methods. However before this method would be very useful for the identification of a dynamic model the parameter space in which convergence will occur must be increased.

The effect of the previously described program changes on convergence was investigated using a second order system not including dead time. Fifty data points were generated as in the previous tests and three significant figures were used. Three initial starting points were chosen so that two, namely 5. and 12. were within the parameter space of convergence for the original general program and the third, 20. was outside. The results of this test are shown in Table 2.3.

Defining the state variables in terms of the changes of the parameters had no effect on the convergence of quasilinearization. This was expected as the same values should have been calculated at each iteration for the two programs (Table 2.3). If it can be assumed that the physical system is at steady state before the forcing
<table>
<thead>
<tr>
<th>Correct Parameters</th>
<th>$c_2$</th>
<th>$c_1$</th>
<th>$g$</th>
<th>Iterations</th>
<th>Converged</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Starting Point

<table>
<thead>
<tr>
<th>General</th>
<th>5.0</th>
<th>5.0</th>
<th>5.0</th>
<th>5</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.0</td>
<td>12.0</td>
<td>12.0</td>
<td></td>
<td>6</td>
<td>Yes</td>
</tr>
<tr>
<td>20.0</td>
<td>20.0</td>
<td>20.0</td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Change</td>
<td>5.0</td>
<td>5.0</td>
<td>5.0</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>12.0</td>
<td>12.0</td>
<td>12.0</td>
<td></td>
<td>6</td>
<td>Yes</td>
</tr>
<tr>
<td>20.0</td>
<td>20.0</td>
<td>20.0</td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>$\mu_1 = \mu_2 = 0$</td>
<td>5.0</td>
<td>5.0</td>
<td>5.0</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>12.0</td>
<td>12.0</td>
<td>12.0</td>
<td></td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Solution-Assume $(\mu_1 = \mu_2 = 0)$</td>
<td>5.0</td>
<td>5.0</td>
<td>5.0</td>
<td>5</td>
<td>Yes</td>
</tr>
<tr>
<td>12.0</td>
<td>12.0</td>
<td>12.0</td>
<td></td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>
function is introduced, the dimensionality of the problem can be reduced. Normally for a second order system, not including dead time, five values of \( \mu \) must be determined corresponding to the output \( x(0) \), its derivatives \( x'(0) \), and the constant parameters \( c_2, c_1, \) and \( G \). If initially the system is at steady state then \( x(0) \) and \( x'(0) \) are known to be zero so only the constant parameters \( c_2, c_1, \) and \( G \) remain to be determined. The results shown in Table 2.1 indicate that lower order systems have a larger parameter space in which convergence will occur. Thus it was expected that a reduction in the dimensionality of the problem would improve convergence. However, Table 2.3 indicates that this was not the case. When the parameters \( \mu_1 \) and \( \mu_2 \) were restricted to a value of zero convergence was not achieved for a starting point of 12., whereas the general program was successful from this starting point. Subroutine ASSUME also was programmed with \( x(0) \) and \( x'(0) \) equal to zero. Thus when subroutine ASSUME, rather than SOLUTION, was used during each iteration to update the output and its derivatives the convergence was the same as for \( \mu_1 = \mu_2 = 0 \).

**Increasing the Parameter Space of Convergence**

Several methods have been mentioned in recent literature for circumventing this problem of convergence. Connelly and Quon (2) have developed the method of data perturbation. Using this technique the data is altered until convergence occurs for the initially assumed solution. This solution is then used as the initial solution with the original data. If convergence does not occur the data is perturbed again and the procedure repeated until convergence occurs without
altering the data. In this work, two other approaches were investigated to increase the parameter space of convergence:

1. Incorporation of Marquardt's method.
2. Reduction of step size.

Quasilinearization as originally proposed uses Newton's method exclusively and the convergence is thus limited to an area near the true solution. The method of steepest descent, on the other hand, is known to work very well in regions far removed from the true solution but very poorly in areas near the solution. Marquardt's method provides for an automatic transition from the method of steepest descent to Newton's method as the minimum of a function is approached. This method can be incorporated into quasilinearization if a least squares criterion is used. By the proper use of steepest descent and Newton's method the parameter space of convergence should be increased.

To minimize a function $f$ Marquardt's method calculates the change $\zeta_i$ in the parameters $\mu_i$ by solving the equation:

$$(Z + \lambda I)\zeta = B$$

where

$Z^{kik} = P^T D$

$P = \frac{\partial f_i}{\partial h_j}$  \hspace{1cm} i = 1, 2, \ldots L \hspace{1cm} j = 1, 2, \ldots K$

$D = P^T (Y - f_0)$

$L = \text{number of data points}$

$K = \text{number of parameters}$

As $\lambda$ is varied from 0 to $\infty$ the search trajectory varies between that of Gauss-Newton and steepest descent. Quasilinearization as originally
programmed uses the same A matrix to calculate the parameters $\mu_j$ by solving the set of equations:

$$Z \bar{\mu} = b$$

By writing this equation in terms of updating the previous values of the parameters $\bar{\mu}_n$

$$Z(\bar{\mu}_n + \bar{c}) = b$$

the least squares criterion used in quasilinearization can be formulated in the form of Marquardt's method:

$$(Z + \lambda I)\bar{c} = b - Z\bar{\mu}_n$$

Using the same test of convergence as was previously discussed the effect of Marquardt's method was investigated. Various starting values of $\lambda$ were tried along with various rates of changing the value of $\lambda$ from iteration to iteration. As the solution is approached the value of $\lambda$ is decreased to take advantage of both steepest descent and Newton's method in the regions they work best. The efficiency of Marquardt's method is thus greatly influenced by both the starting value of $\lambda$ and the rate at which it is varied.

Table 2.4 shows that at a starting point of 20 the most efficient convergence was obtained with a starting $\lambda$ of .01 decreasing it by a factor of 10 during each iteration. At a starting point of 50 convergence would not occur when $\lambda$ was varied between iterations. At this starting point it was found best to hold the $\lambda$ constant at .001. At a starting point of 100 the search trajectory for steepest descent was found to be in the wrong direction. Newton's method moved in the right direction but the step size was too large for convergence to occur.
<table>
<thead>
<tr>
<th>Correct Parameters</th>
<th>2.5</th>
<th>2.5</th>
<th>2.5</th>
<th>5</th>
<th>Yes</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting Point</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marquardt's Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>Yes</td>
<td>0.01</td>
<td>λ/10</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>6</td>
<td>Yes</td>
<td>10</td>
<td>λ/10</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>16</td>
<td>Yes</td>
<td>10</td>
<td>λ/10</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>No</td>
<td>1</td>
<td>λ/2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>25</td>
<td>Yes</td>
<td>0.001</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>38</td>
<td>Yes</td>
<td>0.002</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>No</td>
<td>0.001</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
The problem of determining what value of $\lambda$ should be used and at what rate it should be varied is not peculiar to the incorporation of Marquardt's method into quasilinearization. Even when Marquardt's method is used as an optimization technique by itself the same problem exists. No good general procedure is known for determining the value of $\lambda$. However, this test did demonstrate that Marquardt's method does increase the parameter space of convergence.

Reduction of Step Size

Since the previous test indicated that Newton's method often gave correct search trajectories but that too large a step size was preventing convergence, it would appear advantageous to take only a portion of the calculated step initially, but the whole step in regions near the true solution. Two methods of varying the step size were tested:

1. The step size was varied in a preassigned manner.
2. The step size was adjusted as part of the iteration cycle.

The problem with the first method is that the poorer the assumed solution, the smaller the initial step size must be to assure convergence. Any general function that could be used to adjust the step size for a wide variety of starting points would thus be rather inefficient for starting points close to the true solution. On the other hand, the use of a function that is adjusted for each starting point will require several trials before the best one is found. To demonstrate that this method will work a function was chosen which could be adjusted for various starting points and asymptotically approached 1.
Portion of step size $= 1 - (h)^{-n}$

By adjusting the value of $h$ the rate at which the function approached 1. was changed. The second method which adjusts the step size as part of the iteration cycle does not have this problem and would therefore be more efficient. This method was programmed to check the sum of the squares and if no decrease was noted over the previous iteration the step size was cut in half until a decrease did occur or until the step size was insignificant, at which point the program terminated. It is known this procedure does not guarantee the location of the optimum as the search trajectory for Newton's method occasionally is in a direction in which the criterion function increases. This could have been corrected however by moving in the negative gradient direction when Newton's method failed to show on improvement.

The same test as was discussed previously was used to check the effect of these two methods on convergence. Using the preassigned change in the step size always required several trials before the right function was found. As indicated in Table 2.5 convergence was achieved for starting points of 20, 50, and 100. The second method which calculates the step size internally proved to be very efficient and was successful up to a starting point of 200. For this particular problem the search trajectory of Newton's method was always in a direction in which the criterion function decreased.

**Summary**

The method of quasilinearization has thus been shown to correctly identify the parameters in a dynamic model including dead time up to
<table>
<thead>
<tr>
<th>TABLE 2.5</th>
</tr>
</thead>
</table>

**COMPARISON OF CONVERGENCE USING REDUCED STEP SIZE**

<table>
<thead>
<tr>
<th>Correct Parameters</th>
<th>$c_2$</th>
<th>$c_1$</th>
<th>$g$</th>
<th>Iteration</th>
<th>$h$</th>
<th>Converged</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>2.5</td>
<td>2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Starting Point**

<table>
<thead>
<tr>
<th>Portion of Step Size Reassigned</th>
<th>$c_2$</th>
<th>$c_1$</th>
<th>$g$</th>
<th>Iteration</th>
<th>$h$</th>
<th>Converged</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>11</td>
<td>.8</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>50</td>
<td>13</td>
<td>.91</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>18</td>
<td>.95</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Portion of Step Size Calculated</th>
<th>$c_2$</th>
<th>$c_1$</th>
<th>$g$</th>
<th>Iteration</th>
<th>$h$</th>
<th>Converged</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>6</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>50</td>
<td>6</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>7</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>200</td>
<td>8</td>
<td></td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
third order from input-output data. The accuracy of the answer is affected by the number of significant figures in the data and can be increased by increasing the amount of data. The number of calculations and thus the computer running time for this particular problem can be greatly reduced by defining the state variables in terms of the change in the parameters and by using the model equation instead of the fundamental solutions to update the output and its derivatives. The parameter space in which convergence will occur for this particular problem was demonstrated to be increased by the incorporation of Marquardt's method into quasilinearization or by using only a portion of the step size calculated in quasilinearization. The overall efficiency of this method for this particular problem is thus probably comparable with optimization methods.
LITERATURE CITED


NOMENCLATURE

\( a_i \) = parameters related to the time constants of the system

\( A \) = constant matrix \((m+1 \times m)\)

\( B \) = constant vector \((1 \times m)\)

\( g \) = steady state system gain

\( m \) = \( 2N + 2 \)

\( M(t) \) = level of manipulated variable at time \( t \)

\( n \) = solution number

\( N \) = order of model equation

\( q \) = whole number of sampling periods delay in \( \theta \)

\( X(t) \) = level of system output at time \( t \)

\( Y_{n_i} \) = fundamental solutions of system equations

\( Y_p \) = particular solution of system equations

\( \theta \) = dead time in units of sampling time

\( \delta \) = change in parameters

\( \mu \) = weighting factors for fundamental solutions
CHAPTER III

THEORETICAL DEVELOPMENT OF PREDICTOR MODELS

The problem of interest is described by Figure 3.1. This is a simple closed loop system in which discrete observations of the output are fed back into the computer. Knowing both the output value and the setpoint, the computer calculates the value of the manipulated variable to keep the output at the setpoint. The approach to this problem taken by Box and Jenkins (1,2,3,4,5) is to develop a deterministic model for the system dynamics and a stochastic model for the disturbances. As this model can predict the error at some time in the future, the manipulated variable can be adjusted to compensate.

Stochastic Model

As the objective of a control system is to compensate for disturbances entering the process, a model to predict the unmeasurable disturbances is essential. The model should be simple and yet should describe a wide variety of situations. Although disturbances may enter at any point in the system, the model need only represent their accumulated effect at the output. The disturbance $Z_p$ at time $p$ is therefore defined in terms of the effect it would produce at the output if no control were applied (as if the system were operating open loop). This definition allows the disturbances to be represented as entering subsequent to the system dynamics in Figure 3.1. A general non-stationary model to describe these disturbances is:
Figure 3.1. Description of Computer Control System.
\[ Z_{p+1} = y_{-j} y_{-1}^{-1} \alpha_p + \ldots + y_{-1} \alpha_p + y_0 \alpha_p + y_1 y_2 \alpha_p + \ldots + y_{n-1}^m \alpha_p + \alpha_{p+1} \]  
(3.1)

where

\[ n_o \alpha_p = \alpha_p - \alpha_{p-1} \]

\[ S_o \alpha_p = \alpha_p + \alpha_{p-1} + \alpha_{p-2} + \ldots \]  
(3.2)

and the sequence \( \alpha_{p+1}, \alpha_p, \alpha_{p-1} \) is white noise. Box and Jenkins have found that nearly all disturbances can be satisfactorily described with only two parameters of the above model, namely:

\[ Z_{p+1} = y_0 S_o \alpha_p + y_1 S_o^2 \alpha_p + \alpha_{p+1} \]  
(3.3)

For control purposes, a predictor is needed to predict the disturbance \( Z_{p+t} \) knowing only values up to and including time \( p \). The least squares prediction \( \hat{Z}_p(t) \) of \( Z_{p+t} \) can be obtained from the above model by omitting the \( \alpha \)'s which have not been observed. The one step ahead predictor is therefore:

\[ \hat{Z}_p(1) = y_0 S_o \alpha_p + y_1 S_o^2 \alpha_p \]  
(3.4)

where \( \hat{Z}_p(1) \) denotes the prediction of \( Z_{p+1} \) made at time \( p \). For computational purposes it is more convenient to write the predictor in terms of updating the previous value:

\[ \hat{Z}_p(1) = \hat{Z}_{p-1}(1) + y_0 \alpha_p + y_1 S_o \alpha_p \]  
(3.5)

Specifying that \( Z_{p+1} - \hat{Z}_p(1) = \alpha_{p+1} \)
the disturbance model becomes

\[ Z_{p+1} - Z_p = y_0 \alpha_p + y_1 S_o \alpha_p + \alpha_{p+1} - \alpha_p \]  
(3.6)

This equation will be used to model the characteristics of the disturbances.

**Dynamic Model**

The dynamics (other than dead time) of processes are usually modeled using linear differential equations of the type:
\[ a_n \frac{d^n X(t)}{dt^n} + \ldots + a_1 \frac{dX(t)}{dt} + X(t) = gM(t) \]  \hspace{1cm} (3.7)

where \( g \) is the steady state gain and the \( a \)'s represent the dynamic parameters. In Laplace transform notation this becomes:

\[ \frac{X(S)}{M(S)} = \frac{g}{a_n S^n + \ldots + a_1 S + 1} \]  \hspace{1cm} (3.8)

with all initial conditions being zero. Fortunately, many dynamic systems can be adequately described by including only up to the second derivative term, thus leaving:

\[ a_2 \frac{d^2 X(t)}{dt^2} + a_1 \frac{dX(t)}{dt} + X(t) = gM(t) \]

In a discrete system the differential equation must be replaced by a difference equation. Defining \( X_p \) as the specified value of the output from time \( p \) to time \( p+1 \), it then takes one sampling period for the change in the input to be reflected in the output. The difference equation for the dynamics of the process thus becomes

\[ (1 + \theta_1 \nu + \theta_2 \nu^2) X_{p+1} = gM_p \]  \hspace{1cm} (3.9)

Writing the model in terms of adjustments*:

\[ (1 + \theta_1 \nu + \theta_2 \nu^2) X_{p+1} = gM_p \]  \hspace{1cm} (3.10)

Thus far the model does not represent any delay or dead time (transportation lag). If the dead time is not an integral number of sampling periods delay as shown in Figure 3.2, \( m_p \) is replaced by \( (q + 1 - \lambda)m_{p-q} + (\lambda - q)m_{p-q-1} \). The complete dynamic model is now:

\* In general, capital letters will denote the present values of the variable, whereas lower case letters will denote changes. That is, \( X_{p+1} = X_{p+1} - X_p \).
Figure 3.2. Dead Time $M(p-\lambda)$.
\[(1 + \theta_1 v + \theta_2 v^2) x_{p+1} = g \left[ (q+1-\lambda) m_{p-q} + (\lambda-q) m_{p-q-1} \right] \quad (3.11)\]

Solving this equation for \(x_{p+1}\) makes it more convenient to generate \(x_{p+1}\) from \(m_p\):

\[x_{p+1} = \delta_1 x_p - \delta_2 x_{p-1} + g(1-\delta_1+\delta_2) [(q+1-\lambda) m_{p-q} + (\lambda-q) m_{p-q-1}] \quad (3.12)\]

where

\[\delta_1 = \frac{\theta_1 + 2\theta_2}{1 + \theta_1 \theta_2} \quad \delta_2 = \frac{\theta_2}{1 + \theta_1 \theta_2}\]

The parameters \(\delta_1\) and \(\delta_2\) are related to the time constants of the system \(\tau_1\) and \(\tau_2\) by the equations:

\[\delta_1 = e^{-\tau_1} \quad \delta_2 = e^{-\tau_2} \quad (3.12a)\]

\[-T + \frac{1}{\tau_1} + \frac{1}{\tau_2}\]

where \(T\) = the sampling period.

**Determination of Parameters**

The models that have been derived so far are:

**THE STOCHASTIC NATURE OF THE DISTURBANCE**

\[Z_{p+1} - Z_p = \gamma_0 \alpha_p + \gamma_1 \alpha_p + \alpha_{p+1} - \alpha_p \quad (3.13)\]

**THE DYNAMIC RESPONSE OF THE SYSTEM**

\[x_{p+1} = \delta_1 x_p - \delta_2 x_{p-1} + g(1-\delta_1+\delta_2) [(q+1-\lambda) m_{p-q} + (\lambda-q) m_{p-q-1}] \quad (3.14)\]

The problem is now to find suitable values for the constants \(\gamma_0\), \(\gamma_1\), \(\delta_1\), \(\delta_2\), \(\lambda\), and \(g\). These parameters could be determined from input-output data taken while some form of control is being applied to the process. We must therefore relate the input \(M_p\) to the output
through the models. It can be seen from Figure 3.1 that
\[ Z_p + X_p = C_p \]  
(3.15)
or in terms of adjustments
\[ Z_{p+1} - Z_p + X_{p+1} = C_{p+1} - C_p \]  
(3.16)
Substituting in the stochastic model and solving for \( \alpha_{p+1} \) yields
the relationship
\[ C_{p+1} = C_p + \gamma_o \alpha_p + \gamma_1 S \alpha_p + \alpha_{p+1} - \alpha_p + x_{p+1} \]  
(3.17)
where \( x_{p+1} \) is given by the dynamic model. This gives the necessary
link between input and output. The input data or controller action
is used in the dynamic model to calculate \( x_{p+1} \). This is compared
with the record of the output data and the corresponding difference
is attributed to the disturbance. The difference between the pre­
dicted disturbance and the observed disturbance is \( \alpha_{p+1} \).
\[ Z_{p+1} - Z_p = \alpha_{p+1} \]  
(3.18)
It is desired to determine the parameters so as to minimize this
difference. For a least squares fit the criterion used in the op­
timization technique is to minimize \( \sum \alpha_p^2 \); that is, the one-step-ahead
prediction error is to be minimized. For any set of values of the
parameters and a record of input-output data a set of \( \alpha_p \)'s can be
calculated recursively. The calculation is initiated by setting
\( \alpha_1 = 0, x_0 = 0, x_1 = 0, m_0 = 0 \) and using Equation (3.4) to calculate
\( x_{p+1} \) and Equation (3.17) to calculate \( C_{p+1} \).
The search for the model parameters is aided by knowing the
stability limits of the models:
\[ \lambda \geq 0 \]
\[ \delta_1 < \delta_2 + 1 \]
\[ \delta_1 > -\delta_2 - 1 \]
\[ \delta_2 < 1 \]
\[ g \text{ no limit} \]
\[ \gamma_0 > 0 \]
\[ \gamma_0 < (4 - \gamma) / 2 \]
\[ \gamma > 0 \]

Outside this region the model becomes unstable and leads to very large sums of the squares.

An analysis of the above model equations indicates that an exact fit of process data will only be possible if \( \gamma_0 \) and \( \gamma \) are zero or when the process has pure dynamics and no disturbances. This is inherent in the model for when an exact fit is obtained the one-step-ahead prediction errors (the \( \alpha_p \)'s) must be zero, and the equations reduce to a pure dynamic system. In this way the stochastic model is very similar to the discrete algorithm for a proportional plus integral controller, in that it requires an error to have a non-zero output. It can also be seen that the stochastic model contains a term which is a constant times the current value of the error plus a term containing a constant times the sum of the previous errors, which is the same form as the controller algorithm.

In some cases it may be possible to reduce the complexity of the above models, or it may be necessary to add more terms. Knowing what is involved in a particular process may help to determine how complicated a model is necessary. Various other models can be tested,
retaining in the final model only those terms which contribute significantly to a reduction in the sum of the squares. These models are thus very flexible and can be adjusted to satisfactorily describe a large variety of processes.
LITERATURE CITED


CHAPTER IV
DETERMINING THE MODEL PARAMETERS

Given input-output data from the physical system the model parameters must be determined to minimize the one-step-ahead prediction error (minimize $\sum \varepsilon^2_p$). The approach taken in this chapter is to screen the various available optimization techniques to determine which one works best on this particular problem. The success of the model to predict the error from setpoint depends on accurately determining the model parameters. The dynamic parameters, namely $\delta_1$, $\delta_2$, and $g$ are the most critical. Small errors in any of these parameters may cause the controlled process to be unstable, or at least very poorly controlled. The disturbance parameters $\gamma_0$, $\gamma_1$, on the other hand, are not nearly as critical. For this reason the initial problem considered was that of determining only the four parameters of the dynamic model. Only those techniques which perform satisfactorily during this initial screening are considered for use on the overall problem discussed later in the chapter. This will give some assurance that for the case of the purely deterministic system that the best model parameters will be determined.

When the dynamic model is considered by itself there are other methods of determining the model parameters besides optimization techniques, including regression analysis and linear programming. Due to the deterministic model being in difference form and the fact
that the criterion function is the minimum sum of the squares, the

two parameters related to the time constants, $\delta_1$ and $\delta_2$, and the sys-
tem gain $g$ can be determined by regression analysis while a one-
dimensional search is still required to determine the dead time $\lambda$.

This technique is used in this chapter to reduce the dimensionality
of the search by one and give a different contour surface on which to
test the optimization techniques.

All the optimization methods used in this chapter can be classi-
fied into one of two categories: (1) those which use mathematical
techniques or gradient methods and (2) those which use directed
search techniques or pattern methods. The gradient methods calculate
in some manner the partial derivatives of the criterion function and
use this information to determine in which direction to move. The
methods that are included in this group are: Optimum Gradient,
Gradient Partan, Davidon's methods, Davidon's method with modifica-
tions suggested by Fletcher and Powell, Conjugate Gradients, Mar-
quardt's method, and Rotational Discrimination. The directed search
methods on the other hand do not evaluate the gradient but take trial
steps in various directions, establishing a pattern which dictates
the search direction. Included in this group are: Pattern Search,
Powell's method, and Rosenbrock's Hill Climbing method. A description
of each of the optimization techniques and a discussion of the pecu-
liarities of each of the computer programs is presented in Appendix A.

**Four Parameter--Dynamic Model**

All programs used in this survey were written in subroutine
form in FORTRAN IV and run under the IBSYS monitor for the IBM 7040.
The criterion function contours in the stable region for the dynamic
model have the general shape of a steep sided curved valley. The bottom of the valley and the regions surrounding the valley are quite flat. The stable region for the dynamic model is surrounded by a region yielding very high values of the criterion function and the stable region slopes away from the boundaries. The problem can therefore be treated as an unconstrained optimization as long as the starting point is within the stable region.

The input-output data used in each case was generated from the dynamic model using a step input. The parameter values were therefore known at the minimum and a perfect fit was possible. In order to have a valid comparison between the various techniques the same starting point was used. The value of the criterion function at this point was 5563, the units being the square of the units of the error signal.

When comparing various optimization techniques the number of iterations is not very useful as the connotation for iteration varies from technique to technique. The two most useful numbers for comparison are computer running time and total number of function evaluations. So that the running time would be a valid comparison only the optimal values of the parameters were printed. Also listed in the table of results is the criterion function evaluations to reach a criterion function of 5. This was done in an effort to show that some of the techniques work very nicely as middle-game strategy and very poorly as end-game strategy.

Of the gradient techniques used optimum gradient had the most difficulty locating the minimum. During the initial iterations significant reduction in the criterion function was obtained. As
the method descended into the valley, however, a great deal of oscillation from one side of the valley to the other occurred with only very gradual improvement in the criterion function. A modified form of steepest descent (16) was also tried. In this version the gradient partial derivatives are replaced with normalized derivatives

$$\frac{\partial F}{\partial \ln(X_i)} = \frac{\partial F}{\partial X_i} |X_i|$$

Since $\partial X_i$ and $X_i$ have the same dimensions, the effect of the scale on the derivatives is removed. On this particular test problem this modification showed some improvement over steepest descent in the first few iterations. However, the minimum was never attained using this method.

Marquardt's method for this particular problem behaved very similar to steepest descent, as indicated by the program selecting $\lambda$ values between 9 and 27. Gradient partan showed a major improvement over steepest descent especially in the first portion of the search. Table 4.1 shows that it took optimum gradient 1190 function evaluations to reach a criterion function of 5 while gradient partan took only 232 function evaluations. It therefore would be possible to use this method as a middle-game strategy provided a good end-game was available. Conjugate gradients also showed a need for a good end-game. This method required 215 function evaluations to reach a function evaluation of 5 and another 450 function evaluations to reach a criterion function of .00142. However both of these methods generally operated as well as Rosenbrock's hill climbing method.

Rotational discrimination, Davidon's method, and Davidon's method with modifications suggested by Fletcher and Powell worked equally
<table>
<thead>
<tr>
<th>Method</th>
<th>Function Evaluations C.F.&lt;5</th>
<th>Total Function Evaluations</th>
<th>Computer Time, Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimum Gradient</td>
<td>1190</td>
<td>2600</td>
<td>323</td>
</tr>
<tr>
<td>Marquardt</td>
<td>455</td>
<td>1940</td>
<td>543</td>
</tr>
<tr>
<td>Gradient Partan</td>
<td>232</td>
<td>1118</td>
<td>117</td>
</tr>
<tr>
<td>Rosenbrock II</td>
<td>141</td>
<td>793</td>
<td>96</td>
</tr>
<tr>
<td>Conjugate Gradient</td>
<td>215</td>
<td>665</td>
<td>78</td>
</tr>
<tr>
<td>Rosenbrock I</td>
<td>632</td>
<td>1127</td>
<td>77</td>
</tr>
<tr>
<td>Poof</td>
<td>85</td>
<td>318</td>
<td>38</td>
</tr>
<tr>
<td>Pattern</td>
<td></td>
<td>315</td>
<td>38</td>
</tr>
<tr>
<td>Fletcher Powell</td>
<td>120</td>
<td>210</td>
<td>25</td>
</tr>
<tr>
<td>Powell</td>
<td>48</td>
<td>165</td>
<td>20</td>
</tr>
<tr>
<td>Davidson</td>
<td>120</td>
<td>160</td>
<td>19</td>
</tr>
<tr>
<td>Rot. Discrim.</td>
<td></td>
<td>90</td>
<td>175</td>
</tr>
</tbody>
</table>
well on this problem. Table 4.1 shows that the latter two required a few more function evaluations but have significantly lower running times. Rotational discrimination also has the added problem of determining external scale factors, which are not always easy to calculate. The success of both Davidon's method and the Fletcher Powell modification depend on keeping the Hessian matrix positive definite, which means that the gradients are accurately calculated and that the minimum is accurately located during each linear search. These three methods work better than Rosenbrock's by a factor of 3 to 4 on this particular problem. Rosenbrock's method required 632 function evaluations while Davidon's method required 160, Fletcher Powell modification 210 and Rotational Discrimination 90 function evaluations.

The biggest problem with all pattern type searches is to choose the correct initial step size so that the methods work efficiently. Rosenbrock's Hill Climbing method has a fixed initial step and probably for that reason is less efficient than the other pattern type searches tried. Two different programs for Rosenbrock's method were tried. The program called Rosenbrock's I is called only once from the main program, with the optimum being returned. The program called Rosenbrock's II returns to the main program after each iteration where the check for convergence is made. If convergence is not achieved the search subroutine must be called again. The first of these programs seems to be a little more efficient in its operation.

The initial step size for pattern search was chosen to be one order of magnitude smaller than the estimated range of the variables. Since this method searches only in the coordinate directions it is not
as powerful as methods which calculate new search directions as part of the iteration cycle. It is possible for pattern search to get hung-up on a high resolution ridge, or the method may prematurely decrease its step size and lose its efficient operation due to the ridge. Even with all the shortcomings that can be foreseen for this simple method, it has worked surprisingly well on all the problems tried. Pattern search was about a factor of 2 times better than Rosenbrock's.

Powell's directed search technique (required 165 function evaluations) is as efficient in this case as the three best gradient methods. The initial step size was chosen to be two orders of magnitude smaller than the estimated range of the variables. The program called POOF is Powell's method with an open ended Fibonacci search replacing the one suggested by Powell. This program was less efficient as a specified number of experimental points were tried at each iteration. The effectiveness of the Fibonacci search depends on being able to bracket the minimum in the first three experimental points. If this is not done much of its efficiency is lost.

Reduction of Dimensionality

Due to the nature of the equations used for the dynamic model and the fact that the criterion function is expressed in terms of the sum of the squares, the dimensionality of the search problem can be reduced. Since the gain enters the equation in a linear fashion it can be determined by a linear regression, while the other three parameters are still determined by a nonlinear search. Advantages may be gained from this in not only reducing the dimensionality of the search but also in changing the shape of the contour
surfaces. It is known that the gain caused the steep-sided valley in the original problem. By determining this parameter by regression analysis the contour surface should have more nearly symmetrical contours, and therefore be a less difficult problem for the search techniques to handle.

To implement this technique the criterion function was evaluated in the following way:

1. The process input values $X_p$ and the three parameters $B$ from the search subroutine were read in.
2. The output $C_p$ was generated from the model equation with a gain of 1.
3. The gain was then determined as follows by minimizing

$$f = \sum (C_p - C_{pOBS})^2$$

using a linear regression:

$$C_p = g \varphi (B)$$

$$f = \sum (C_p - C_{pOBS})^2$$

$$= \sum (g \varphi (B) - C_{pOBS})^2$$

$$\frac{\partial f}{\partial g} = 2 \sum (g \varphi (B) - C_{pOBS}) \varphi (B) = 0$$

$$0 = g \sum \varphi (B)^2 - \sum \varphi (B) C_{pOBS}$$

$$g = \frac{\Sigma \varphi (B) C_{pOBS}}{\Sigma \varphi (B)}$$

Using the same input-output data that was used in the four parameter search, the optimization techniques were evaluated using this form of the criterion function.

The results of this study are shown in Table 4.2. Due to the change in contour shape some of the methods showed improved efficiency over the previous case and some were less efficient. Optimum gradient
<table>
<thead>
<tr>
<th>Method</th>
<th>50 Data Points</th>
<th>Function Evaluations C.F.&lt;5</th>
<th>Total Function Evaluations</th>
<th>Computer Time Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient Partan</td>
<td></td>
<td>811</td>
<td></td>
<td>83</td>
</tr>
<tr>
<td>Rosenbrock II</td>
<td>98</td>
<td>516</td>
<td></td>
<td>62</td>
</tr>
<tr>
<td>Optimum Gradient</td>
<td>330</td>
<td>459</td>
<td></td>
<td>55</td>
</tr>
<tr>
<td>Rosenbrock I</td>
<td></td>
<td>402</td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>Poof</td>
<td>145</td>
<td>308</td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>Fletcher Powell</td>
<td>116</td>
<td>280</td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>Davidon</td>
<td></td>
<td>276</td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>Pattern</td>
<td></td>
<td>237</td>
<td></td>
<td>29</td>
</tr>
<tr>
<td>Conjugate Gradient</td>
<td>68</td>
<td>216</td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>Powell</td>
<td></td>
<td>166</td>
<td></td>
<td>21</td>
</tr>
</tbody>
</table>
and conjugate gradients showed the biggest improvement. Optimum gradient required 2699 function evaluations for the four parameter search and only 459 using this technique, while conjugate gradients required 665 function evaluations for the four parameter search and only 216 using this technique. All of the techniques showed some reduction in function evaluations except for Davidon's method and the Fletcher-Powell modification of Davidon's.

This survey of various optimization techniques was conducted to determine the method which is most suited for this particular problem. Since the end product of this research would be used to control a real physical system, it is important to have some assurance that the optimization method used will determine the correct model parameters. The efficiency of the method is also important as we anticipate that there will not be much computer time available on a small control computer to perform such calculations. The optimization program should also be simple so that it will not require a large block of computer core storage. Using these criterion some of the optimization techniques can be eliminated. Rosenbrock's Hill Climbing method, Conjugate Gradients, Gradient Partan, Marquardt's method, and Optimum Gradient were eliminated as they require a significantly larger number of function evaluations than the other methods. Rotational Discrimination was eliminated due to the length of the program and the long running time. Remaining for further testing on the six parameter model are Pattern, Fletcher Powell modifications of Davidon's method, Davidon's method, and Powell's method.

An effort was made on all of the programs to run them in the most
efficient manner. For instance, a number of step sizes were tried on the directed search techniques and it was found to be most efficient to decrease the step size as the minimum is approached, rather than to use a constant step size. Although some of these alterations greatly increased the efficiency of the method on this particular problem they may not be applicable to all problems. In general there was a problem at this point with all the methods in determining when to terminate the search. Several different techniques were tried but a completely satisfactory method was not found. This problem was not further investigated, however, as the object of this survey was just to screen available techniques to determine which methods should be tried on the six parameter model.

Six Parameter--Complete Model

The results from the screening of various optimization techniques on the pure dynamic system \((\delta_1, \delta_2, \lambda, g)\) indicate that only four of the methods, namely, Davidon, Fletcher-Powell, Powell, and Pattern, remain for consideration on the six parameter model. The data that was used for this test was obtained by forcing the dynamic model with a unit step and superimposing partially correlated noise on the output. Using this technique the correct dynamic parameters, namely \(\delta_1, \delta_2, g\) and \(\lambda\), were known but a perfect fit was not possible due to the noise. The noise was generated by passing white noise through a first order lag and adjusting the gain so that a desired value of the root mean square (RMS) was obtained at the output. The time constant and RMS were chosen so that the data looked very similar to actual process data. The results of this test are shown in Table 4.3.
<table>
<thead>
<tr>
<th>Method</th>
<th>Total Function Evaluations</th>
<th>Computer Time Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fletcher Powell</td>
<td>623</td>
<td>268</td>
</tr>
<tr>
<td>Pattern</td>
<td>454</td>
<td>216</td>
</tr>
<tr>
<td>Powell</td>
<td>342</td>
<td>160</td>
</tr>
<tr>
<td>Davidon</td>
<td>168</td>
<td>74</td>
</tr>
</tbody>
</table>
The value of the criterion function at the starting point was 81 while the best fit obtained had a criterion function of 1.7. Since 200 data points were used instead of 50, as was used in Tables 4.1 and 4.2, each function evaluation took four times as long.

It appears from Table 4.3 that the best method to use on this particular problem would be Davidon's; however, other things must be considered. The starting point of the search seemed to have a great effect on the operation of each method. At some starting points the optimization methods would terminate indicating that a minimum was found with a low value of the criterion function, but the dynamic parameters would be far removed from what was known to be the correct parameters. In certain instances the gradient techniques would terminate indicating a minimum was found but the criterion function would be very large. This indicated that the reliability of the four optimization techniques must again be considered as well as the efficiency of the method.

To aid in the final decision of which optimum seeking method would finally be used, the contour surfaces were investigated for various forcing functions and various root mean square values of noise. Two parameters were investigated at a time while the others were held at their optimal values for the particular set of data used. It was hoped that this investigation would indicate whether the optimization techniques were truly locating the minimum, why the correct dynamic parameters were not always located and why in some cases the gradient methods did not work at all.

The contour surfaces indicated that when the optimization techniques were terminating with low values of the criterion function that
indeed, some sort of minimum had been located. Since the optimization techniques are working correctly the problem must be with the shape of the contour surface. The deviation of the dynamic parameters from the known correct values can be accounted for by three situations: 1. The possible presence of local minimums, 2. The flattening of the contours of the dynamic parameters as the amount of noise is increased and 3. A minimum did not exist at the known dynamic parameters.

The two dimensional contours for the dynamic parameters \((\delta_1, \delta_2, g, \lambda)\) indicate in some cases the possible presence of local minimums. For example, in Figure 4.1 there appear three unconnected areas displaying the lowest value of the criterion function in the vicinity. This same phenomena is also seen in Figure 4.4. It is not correct to conclude from these figures that these are local minimum, but only that there is the possibility, as only two of the six dimensions were considered. The presence of ridges in the contour surface could also give the appearance of Figure 4.1 in two dimensions. The X's on the diagrams indicate the termination points of the search techniques. In all cases, when the termination point of the optimum seeking method and the contour surfaces were compared, there was no indication of the presence of local minimum. If local minimum were present in the vicinity of the global optimum the search would have most probably terminated at different final values for the various starting points used. None of the optimization methods are guaranteed to find the global optimum in the presence of local minimum. If indeed this was the problem, the only solution is to start the search at various starting points and accept the best minimum found as the global optimum.
Figure 4.1. Two Dimensional Contour $\delta_2$ vs $\delta_1$ (Noise RMS = .1)
Figure 4.2. Two Dimensional Contour $\lambda$ vs $g$ (Noise RMS = .1)
Figure 4.3. Two Dimensional Contour $\delta_2$ vs $\delta_1$ (Noise RMS = 0.2)
Figure 4.4. Two Dimensional Contour $\lambda$ vs $g$ (Noise RMS = .2)
As the amount of noise added to the system is increased, relative to the forcing function, the contour surfaces for the dynamic parameters flatten out, and the gradients for the stochastic parameters increase. The flattening of the contours can be seen by comparing Figures 4.1 and 4.3 and Figures 4.2 and 4.4. The RMS value of the noise in Figures 4.1 and 4.2 was .1 while in Figures 4.3 and 4.4 it was .2. As the RMS value of the noise increased the gradient of the parameters $\delta_1$ and $\delta_2$ decreased by about two orders of magnitude. In the case of the gain $g$ the gradient decreased by 4 orders of magnitude. At the same time the gradients of the stochastic parameters ($\gamma_0$, $\gamma_1$) both increased by one order of magnitude. Accurate dynamic parameters are therefore difficult to obtain when the amount of noise is large relative to the forcing function. This would indicate that it may be advantageous to take two sets of process data when trying to fit this model: one with no forcing function and one with a relatively large forcing function if possible. The first set of data would have steep gradients with respect to the stochastic parameters so $\gamma_0$, $\gamma_1$ could be accurately located. Holding the stochastic parameters at their optimal values the second set of data with steep gradients with respect to the dynamic parameters would be used to determine $\delta_1$, $\delta_2$, $\lambda$ and $g$.

The third problem associated with obtaining incorrect dynamic parameters but low values of the criterion function is that minimums in some cases do not exist at the correct dynamic parameters. This problem seemed to be most prevalent with the dead time but did affect the other dynamic parameters as the amount of noise was increased.
Figures 4.2 and 4.4 most clearly demonstrate this problem. The correct value for the dead time is .8 (indicated in Figures 4.4 by P) but the two dimensional contours display a ridge in this area with the optimal value at a dead time of 3. The X indicated the final value located during the search, indicating again that the optimization methods are working correctly. The problem is the contour surface does not have the correct shape. Two ways of possibly changing the contour surface is to change the shape of the forcing function used to obtain the information and increasing the number of data points used to create the surface.

Three different forcing functions were chosen for testing based on their ease of generation in the actual process situation. They included 1. the unit step, 2. the unit pulse, 3. the unit pulse occurring at random intervals, and of random duration. Table 4.4 shows the results of these various forcing functions as they effected the two parameters \( \delta_1 \) and \( \delta_2 \). Using a unit step forcing function the parameters \( \delta_1 \) and \( \delta_2 \) could be correctly identified up to a root mean square value of noise of .1, for a unit pulse up to .3 and for a random unit pulse up to .5 or slightly higher. By exciting the dynamic portion of the system more frequently more information on the dynamic parameters is incorporated into the contour surface. The random pulse was far superior to the other forcing function tested and was used for all subsequent experimental work. Table 4.5 indicated the effect of increasing the number of data points on \( \delta_1 \) and \( \delta_2 \) which again is just increasing the amount of information used in the contour surface. Economically it is desirable to use the fewest data
<table>
<thead>
<tr>
<th>FORCING FUNCTION</th>
<th>NUMBER OF DATA POINTS</th>
<th>RMS OF NOISE</th>
<th>$\delta_1$</th>
<th>$\delta_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit Step</td>
<td>200</td>
<td>.1</td>
<td>1.506</td>
<td>.5625</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>.2</td>
<td>1.291</td>
<td>.3915</td>
</tr>
<tr>
<td>Unit Pulse</td>
<td>200</td>
<td>.3</td>
<td>1.504</td>
<td>.5653</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>.5</td>
<td>1.449</td>
<td>.5653</td>
</tr>
<tr>
<td>Random Unit Pulse</td>
<td>200</td>
<td>.5</td>
<td>1.507</td>
<td>.5625</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>.8</td>
<td>1.519</td>
<td>.5615</td>
</tr>
<tr>
<td>Correct Parameters</td>
<td></td>
<td></td>
<td>1.500</td>
<td>.5625</td>
</tr>
<tr>
<td>FORCING FUNCTION</td>
<td>NUMBER OF DATA POINTS</td>
<td>RMS OF NOISE</td>
<td>$\delta_1$</td>
<td>$\delta_2$</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------</td>
<td>--------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>Random Unit Pulse</td>
<td>50</td>
<td>.2</td>
<td>1.835</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>.2</td>
<td>1.599</td>
<td>.7002</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>.2</td>
<td>1.5161</td>
<td>.5625</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>.2</td>
<td>1.5076</td>
<td>.5625</td>
</tr>
<tr>
<td>Correct Parameters</td>
<td></td>
<td></td>
<td>1.5000</td>
<td>.5625</td>
</tr>
</tbody>
</table>
points possible without significantly reducing the accuracy of the parameters. Table 4.5 shows that as more data points are used the additional information gained is diminished. For this particular problem it was found that the necessary number of data points to obtain accurate values for $\delta_1$ and $\delta_2$ increased about linearly with respect to the root mean square of the noise (Figure 4.5) for a random pulse forcing function. While changing the forcing function and increasing the number of data points did make it possible to obtain accurate values for $\delta_1$ and $\delta_2$ it had very little effect on the dead time $\lambda$. The dead time $\lambda$ and the process gain $g$ are by far the most difficult parameters to determine in a noisy system. Fortunately the controller is not very sensitive to the dead time so an accurate value will not be required.

When the gradient optimization methods would terminate with very large values of the criterion function, the parameters would be in the unstable region. A short ways into the unstable region the value of the criterion function becomes large enough to overflow the computer. This made it impossible to calculate the gradients at this point and caused the program to terminate. The stability limit that was most generally violated was:

$$\delta_1 < \delta_2 + 1$$

Investigation of the parameters $\delta_1$ and $\delta_2$ shows that they will always lie close to the unstable region as long as the time constants of the system are greater than the sampling period. (Table 4.6) As the time constants of the systems increases with respect to the sampling period the value of $\delta_1$ approaches 2 and the value of $\delta_2$ approaches 1.
Figure 4.5. Number of Necessary Data Points vs RMS Value of Noise.

Random Unit Pulse
TABLE 4.6

COMPARISON OF SYSTEM TIME CONSTANTS WITH $\delta_1$ AND $\delta_2$

<table>
<thead>
<tr>
<th>$T/\tau_1$</th>
<th>$T/\tau_2$</th>
<th>$\delta_1$</th>
<th>$\delta_2$</th>
<th>$\delta_1 - \delta_2^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>0.0270</td>
<td>0.0034</td>
<td>0.0267</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1.0000</td>
<td>0.0025</td>
<td>0.0975</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0.2720</td>
<td>0.0185</td>
<td>0.2565</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0.736</td>
<td>0.136</td>
<td>0.6000</td>
</tr>
<tr>
<td>0.7</td>
<td>0.7</td>
<td>0.996</td>
<td>0.248</td>
<td>0.748</td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td>1.216</td>
<td>0.368</td>
<td>0.848</td>
</tr>
<tr>
<td>0.3</td>
<td>0.3</td>
<td>1.484</td>
<td>0.550</td>
<td>0.934</td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>1.810</td>
<td>0.819</td>
<td>0.991</td>
</tr>
<tr>
<td>0.05</td>
<td>0.05</td>
<td>1.902</td>
<td>0.905</td>
<td>0.997</td>
</tr>
</tbody>
</table>

| 0.1        | 1          | 1.273      | 0.334      | 0.939                    |
| 0.1        | 0.7        | 1.403      | 0.450      | 0.953                    |
| 0.1        | 0.5        | 1.513      | 0.550      | 0.963                    |
| 0.1        | 0.3        | 1.647      | 0.671      | 0.976                    |

* System Unstable if $\delta_1 - \delta_2 = 1.0$. 
The value of the criterion function outside of the stable region increases very rapidly. Both Davidon's and Fletcher Powell's method calculate a step size, make the step, and must then be able to accurately calculate the gradient at this point. The gradient for these two parameters are so much greater outside the stable region than within that a small error in the step size will make it impossible to calculate the gradients.

Pattern Search also had some difficulty with this stability limit for the particular starting point and the step size chosen. If the starting point and the step size were chosen such that it is easy to obtain a difference of exactly 1.0 between the parameters $\delta_1$ and $\delta_2$ then the search would get "hung up" on the boundary. This is illustrated in Figure 4.1, where S is the best point found to date and the t's are the trial steps about it. It is easy to see that all of the trial steps could give higher values of the criterion function. This is one of the inherent problems with Pattern Search, that it can get stuck on very high resolution ridges and valleys. This problem may not stall the method in all situations but the step size most likely will reduce prematurely causing a loss in efficiency. Proper choice of the initial step size seems to circumvent this problem. If the starting point for the search was chosen to be even numbers an initial step size of 1.0 would occasionally cause the method to "hang up" but a set size of .95 always worked with no problems. In general the pattern type searches did not have the problem the gradients method exhibit in the unstable region. Since only the value of the criterion function at any point must be evaluated and
it is higher in the unstable region, the pattern type searches are forced back into the stable region.

For the particular set of data used there also seemed to be a tendency to slightly violate the boundary condition $\gamma_1 \geq 0$. The minimum value of the criterion function for this parameter was always slightly negative, but never greater than $-0.2$. This was found to be quite serious as a negative value for $\gamma_1$ makes the controller unstable. To avoid this necessitated placing constraints on all of the optimization methods. The only search method in which this was satisfactorily accomplished was Pattern Search. Any point that would violate the boundary conditions was treated as though it had a large value of the criterion function even though it was not evaluated. The methods of Powell, Davidon, and Fletcher-Powell were not designed to recognize constraints. An attempt was made to adjust the step sizes of these methods when they went out of bounds but this leads to situations where the optimization method will continually want to step out of bounds.

The original criterion for choosing an optimization method used previously when screening the 4 parameter model must now be applied again to the complete 6 parameter model. The reliability of the final optimization method chosen is very important. To be able to successfully control a process better than the present control schemes will require accurate values of the parameters. If any on-line updating of the model is necessary the ability of the optimization method to locate the correct parameters without getting "hung-up" is essential. Efficiency is also important but not nearly as much as
reliability. Of the four methods tested these criteria limit the final decision to only Pattern Search. For this particular problem where the values of the parameters are limited to very narrow ranges Pattern Search is ideally suited. For the initial identification of the model parameters Pattern Search may take a few minutes more of computer time if very poor guesses of the parameters are made, but will reliably reach a minimum. However, initially a good deal of time can be afforded to locate the best model parameters. If updating of the model is necessary, a very good guess of the model parameters is available, so Pattern Search will be as efficient as any of the other methods. This method also has no difficulty recognizing the boundary condition that was found necessary in the six parameter model. Of all the optimization programs tested Pattern Search contained the fewest number of statements and will therefore require the least computer core storage.

This chapter has described the first phase of this investigation of the development and application of stochastic models for process control. Using only the deterministic portion or the model an initial screening of optimization techniques was made, eliminating all but four methods which were tested on the complete six parameter model. Due to the necessity of having to add one constraint to the optimization and based on the reliability of the method Pattern Search was chosen as the final method to be used.

Summary

To illustrate the type of data on which Pattern Search was tested three systems were chosen which varied only in their relative
amount of noise to the forcing function. The forcing function was a random unit pulse with random duration and the RMS value of the noise was set at .2, .5 and 1. At an RMS value of .2 only 100 data points were required to obtain the correct parameters and a criterion function value of 1.7 (Figure 4.6). At an RMS value of .5, 200 data points were required and the final criterion function was 10.2 (Figure 4.7). While at an RMS of 1., 400 data points were required and the final value of the criterion function was 40.5 (Figure 4.8). All of these values of the criterion function are based on 200 data points. In each of the three cases the value of the stochastic parameters remained the same. This lends support to the earlier suggestion to use two different sets of data when fitting the model. One with no forcing function to determine the stochastic parameters and one with a relatively large forcing function to determine the dynamic parameters. Possibly this would require only as many data points as would be needed for one set of data. If the use of a relatively large forcing function is not possible due to the process operating limits, then a random pulse of maximum allowable height is recommend as the forcing function. Since the stochastic parameters are not a function of the amount of noise the accuracy of the model to predict the disturbances decreases as the amount of noise increases.

The interest in optimization was only to find the most efficient and reliable technique for use on this particular problem. No specific method is recommended as a universal technique to be used on all problems, as it is not felt this is possible. This can be seen by comparing Tables 4.1, 4.2 and 4.3 where the order of efficiency
Figure 4.6. Model Error on Experimental Data (Noise RMS = .2).
Figure 4.7. Model Error on Experimental Data (Noise RMS = .5).
Figure 4.8. Model Error on Experimental Data (Noise RMS = 1.)
changed quite markedly for the three contour surfaces tested. It is also felt that the only way to determine which method will work best on any specific problem is to try them.
LITERATURE CITED


CHAPTER V
DEVELOPMENT OF STOCHASTIC CONTROL

Having developed a procedure which would determine the six parameters in the proposed model from input-output data, an optimal control scheme must be designed. The stochastic portion of the model can be used to predict the error from set point at some number of sampling periods in the future. The change in the manipulated variable that will offset this predicted error is then calculated from the dynamic portion of the model. Included in this chapter are two illustrative examples of the application of this proposed control strategy. The first example is a second order system for which the correct model parameters are known. This system was used to determine the sensitivity of the model and control parameters as well as to investigate the problem of large dead time. A comparison was made between the predictor controller and an optimally tuned proportional plus integral controller for (1) a noisy system, (2) a noisy system with a step change in disturbance, and (3) a noisy system with a step change in set point. The second example is the control of a nonlinear chemical reactor. The optimum seeking method discussed in Chapter IV were used to determine the model parameters. A comparison was also made in this example of an optimally tuned proportional plus integral controller and the experimentally determined predictor controller.

Development of Control Strategy

Using the stochastic model, the best prediction of the deviation
from set point \( t \) sampling periods in advance if nothing is done to prevent it is:

\[
\frac{\dot{z}_p(t)}{\dot{z}_{p-1}(t)} = [\gamma_0 + (t-1)\gamma_1]A_p + \gamma_1 S \alpha_p
\]

5.1

To prevent the system from being off target in the \( p+t \) sampling period by this predicted amount, the manipulated variable \( m_p \) must be adjusted such that the output of the dynamic model \( x_{p+t} \) cancels this predicted error:

\[
-x_{p+t} = \frac{\dot{z}_p(t)}{\dot{z}_{p-1}(t)}
\]

5.2

Using the dynamic model written for the \( p+t \) sampling period the adjustment which will give the correction \(-x_{p+t}\) is calculated:

\[
m_{p-q+t-1} = -A_1 m_{p-q+t-2} + A_2 x_{p+t} - A_3 x_{p+t-1} + A_4 x_{p+t-2}
\]

5.3

where

\[
A_1 = \frac{\lambda-q}{q+1-\lambda} \quad A_2 = \frac{1}{\delta(1-\delta_1+\delta_2)(q+1-\lambda)} \quad A_3 = \delta_1 A_2 \quad A_4 = \delta_2 A_2
\]

For equation (5.3) to be correct \( t \) must be chosen to satisfy the relationship:

\[
t = q + 1
\]

where \( q \) equals the whole number of sampling periods delay in \( \lambda \).

Thus a system with less than one sampling period delay requires a one step ahead predictor and a system with 1 to 2 sampling periods delay requires a two step ahead predictor. If possible the sampling interval should be adjusted so that a one step ahead predictor is used as when \( t \) increases the accuracy of the predictor decreases.

The general control Equation 5.3 calls for a change in the manipulated variable which is a linear function of the previous
manipulated variable, the predicted error from set point, and two previous values of the predicted error. The control computer would therefore have to store Equations 5.1 and 5.3 plus one value of the manipulated variable and two values of the previous predicted error for each control loop.

**Application of Preditor Controller to a Second Order System**

In this first example the system to be controlled is a second order lag including dead time on whose output has been superimposed partially correlated noise with an RMS value of 0.2. The correct dynamic parameters were thus known and the stochastic parameters were found to be

\[ \gamma_0 = 1.6732 \]

and

\[ \gamma_1 = 0. \]

by using Pattern Search. The sensitivity of the model parameters \((\lambda, \delta_1, \delta_2, \theta_0, \gamma_0)\) as well as the control parameters \((A_1, A_2, A_3, A_4)\) were investigated on this system for errors up to \(\pm 20\%\). The sum of the error from the set point squared \((\Sigma e^2)\) and the sum of the change in the manipulated variable squared \((\Sigma m^2)\) were used as the criteria for comparing the effectiveness of the controller. Figure 5.1 shows the effect of the parameter error on the control error. For this particular system the values of \(\delta_1\) and \(\delta_2\) are very close to the unstable region so small errors in this direction makes control impossible. However, even within the stable region these parameters are sensitive only to errors of greater than 15% in one direction. As the dead time is the hardest parameter to determine
The Sum of the Error From the Set Point Squared ($\sum \epsilon^2$)

\[ \gamma_1 = 0 \]
\[ \gamma_0 = 1.6732 \]
\[ \lambda = 0.2 \]
\[ \delta_1 = 1.50 \]
\[ \delta_2 = 0.5625 \]
\[ g = 1.25 \]

Figure 5.1. Sensitivity of Control to Model Parameters.
The Sum of the Change in the Manipulated Variable Squared ($E_m^2$)

Figure 5.2. Effect of Model Parameter Error on Manipulated Variable.

- $\gamma_1 = 0$
- $\gamma_0 = 1.6732$
- $\lambda = 0.2$
- $\delta_1 = 1.5$
- $\delta_2 = 0.5625$
- $g = 1.25$
accurately, the control error is fortunately relatively insensitive to this parameter.

Figure 5.2 shows the effect of the parameter errors on the change in the manipulated variable. If the value of the gain is below the true value a larger change in the manipulated variable will be calculated by Equation 5.3 to obtain the desired output. This explains the larger change in the manipulated variable and poorer control for negative errors in the system gain. A positive error in the dead time means the controller has less time to correct for the predicted error and too large a change in the manipulated variable will result. A value of \( \gamma_0 \) above the true value will predict too high an error and will also cause too large a change in the manipulated variable. Since the parameters \( \delta_1 \) and \( \delta_2 \) are both functions of the two time constants of the system their behavior is not easily explained. However, it is apparent that increasing one causes the same effect as decreasing the other.

Comparing Figures 5.1 and 5.2 it appears that errors of less than 20% in the parameters \( g, \lambda \), and \( \gamma_0 \) that tend to decrease the change in the manipulated variable have very little effect on the control error. Using this fact the change in manipulated variable can be greatly reduced without adversely affecting the control error. For example, in this particular system if the gain is increased by 10% over its true value, \( \Sigma m^2 \) is decreased by 29% and the control error \( \Sigma e^2 \) is only increased by 5%.

The control parameters \( A_1, A_2, A_3, A_4 \) appear to be less sensitive than the individual model parameters (Figures 5.3 and 5.4). The reason for this is that an error in any of the model parameters affects at least three of the control parameters. Since the model
The Sum of the Error From the Set Point Squared ($\sum e^2$)

Per Cent Change

Figure 5.3. Sensitivity of Control to Controller Parameters.

\[
\begin{align*}
A_1 &= 0.25 \\
A_2 &= 16. \\
A_3 &= 24. \\
A_4 &= 9. 
\end{align*}
\]
Figure 5.4. Effect of Control Parameter Error on Manipulated Variable.
parameters are the end product of the system identification problem (from which the control parameters must be calculated), the sensitivity of the control parameters do not have much physical meaning or use.

Since there will be some error in the model parameters, on-line tuning of the controller may be beneficial even though not often practical. It was found that for this particular system where the dynamic parameters were accurately known and where the only error must be in the stochastic parameter \( \gamma_0 \), tuning the controller using the \( \Sigma e^2 \) as the criterion function resulted in the reduction of \( \Sigma e^2 \) by 16% and \( \Sigma m^2 \) by 20%. Since (1) the model parameters have larger gradients and (2) their physical significance would help to determine in which direction to change the parameters, they should be used to tune the controller rather than the control parameters. As relatively small changes in the parameters result in significantly better control it may be possible to use some sort of evolutionary procedure to tune the controller. In certain instances it may be desirable to change the function being minimized from \( \Sigma e^2 \) to \( \Sigma (e^2 + rm^2) \), thus including a penalty for changing the manipulated variable. As shown in Table 5.1 as the value of \( r \) is increased to .01 the control error \( (\Sigma e^2) \) doubles but the change in manipulated variable \( (\Sigma m^2) \) is decreased by a factor of four.

As the dead time \( \lambda \) in the controller Equation 5.3 approaches one for a one step ahead predictor, the control parameters \( A_1, A_2, A_3, \) and \( A_4 \) approach infinity, and thus the changes in the manipulated variable become very large. The reason for this is that the dead
<table>
<thead>
<tr>
<th>Function Minimized</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$A_4$</th>
<th>$\Sigma e^2$</th>
<th>$\Sigma m^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untuned</td>
<td>0</td>
<td>12.8</td>
<td>19.2</td>
<td>7.2</td>
<td>1.724</td>
<td>2813</td>
</tr>
<tr>
<td>$\Sigma e^2$</td>
<td>0.0135</td>
<td>11.245</td>
<td>19.955</td>
<td>8.659</td>
<td>1.435</td>
<td>2247</td>
</tr>
<tr>
<td>$\Sigma e^2+rm^2$</td>
<td>0.0034</td>
<td>11.143</td>
<td>19.757</td>
<td>8.552</td>
<td>1.435</td>
<td>2247</td>
</tr>
<tr>
<td>$r = 0.0001$</td>
<td>0.00135</td>
<td>9.203</td>
<td>18.116</td>
<td>8.935</td>
<td>1.710</td>
<td>1463</td>
</tr>
<tr>
<td>$r = 0.01$</td>
<td>0.4016</td>
<td>6.793</td>
<td>10.286</td>
<td>3.585</td>
<td>2.622</td>
<td>518</td>
</tr>
</tbody>
</table>
time reduces the effective time in each sampling period in which the controller can respond. Thus as the dead time increases the control action must also increase so that the predicted error will be compensated for at the end of the sampling period. For this particular second order system effective control could be maintained up to a dead time of .5. (Table 5.2). However, practically speaking a dead time of .3 was limiting due to the rapid increase in the change of the manipulated variable above this point (Figure 5.5). For systems with too large a dead time the sampling rate must be adjusted such that the dead time falls within the region where control can be achieved without excessive changes in the manipulated variable.

By increasing the sampling rate the dead time can be adjusted so that a multi-step ahead predictor is required. This was found not to be at all successful for this particular system. The additional error encountered by predicting more than one sampling period in advance made the control of the system impossible. Even if control would have been possible, there is often not much point in sampling at rates more frequent than the dead time as little improvement in control can be obtained. The other alternative was to sample less frequently so that a one step ahead predictor could be used. This means that either the model parameters already determined have to be adjusted for the slower sampling rate or a set of data at the slower sampling rate must searched to determine the new parameters. The dynamic parameters can be easily adjusted for any sampling rate once they are known. The gain \( g \) is not a function of the sampling rate, the dead time \( \theta \) has the units of sampling periods so it can be easily adjusted and the parameters \( \delta_1 \) and \( \delta_2 \) can be calculated using
<table>
<thead>
<tr>
<th>Dead Time</th>
<th>$\Sigma \hat{e}^2$</th>
<th>$\Sigma \hat{u}^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.724</td>
<td>2813</td>
</tr>
<tr>
<td>.1</td>
<td>1.724</td>
<td>3966</td>
</tr>
<tr>
<td>.2</td>
<td>1.724</td>
<td>5997</td>
</tr>
<tr>
<td>.3</td>
<td>1.724</td>
<td>10140</td>
</tr>
<tr>
<td>.4</td>
<td>1.724</td>
<td>21487</td>
</tr>
<tr>
<td>.5</td>
<td>1.724</td>
<td>670327</td>
</tr>
<tr>
<td>.6</td>
<td>Unstable</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.5. Change in Manipulate Variable as the Dead Time Increases.
Equation 3.12a. Box and Jenkins (1,2,3,4,5) show that the stochastic parameters for a system whose disturbance can be predicted with some non-zero value of $\gamma_0$ and $\gamma_1 = 0$ can be adjusted for various sampling rates. If only every $h$th observation had been recorded, the resulting disturbance could be predicted using a similar predictor with $\gamma_0$, some function of $h$ and $\gamma_1$ still zero.

$$\frac{\gamma_0^2(h)}{1-\gamma_0(h)} = \frac{h\gamma_0^2}{1-\gamma_0}$$

For systems where $\gamma_1$ is not zero the model parameters for different sampling rates must be obtained from experimental data taken at that sampling rate.

Using a second order system with a dead time of .4 the effect of less frequent sampling on control was investigated. Even when the sampling interval is larger than the dead time, the control achieved using a certain sampling interval may be unnecessarily tight so that less frequent sampling is called for. However, for this particular system reducing the frequency of sampling greatly reduced the effectiveness of the controller (Table 5.3).

Today in the chemical industry the most widely accepted and often used form of control is a proportional plus integral (PI) controller. For this reason the discrete form of a PI controller will be used as the standard to which the performance of the preditor controller will be compared. The system was a second order lag with no dead time on which partially correlated noise with an RMS value of .2 was superimposed. A comparison between the two controllers was made for (1) the noisy system, (2) the noisy system with a step
TABLE 5.3
THE EFFECT OF LESS FREQUENT SAMPLING

<table>
<thead>
<tr>
<th>h</th>
<th>$A_1$</th>
<th>$A_3$</th>
<th>$A_4$</th>
<th>$\Sigma e^2$</th>
<th>$\Sigma m^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6667</td>
<td>21.33</td>
<td>32.0</td>
<td>12.0</td>
<td>1.724</td>
<td>21487</td>
</tr>
<tr>
<td>2</td>
<td>5.25</td>
<td>5.854</td>
<td>1.646</td>
<td>7.756</td>
<td>320</td>
</tr>
<tr>
<td>3</td>
<td>2.875</td>
<td>2.427</td>
<td>0.511</td>
<td>9.783</td>
<td>58</td>
</tr>
<tr>
<td>4</td>
<td>2.415</td>
<td>1.768</td>
<td>0.241</td>
<td>13.471</td>
<td>47</td>
</tr>
</tbody>
</table>
change in disturbance of 1. and 10. and (3) the noisy system with a step change in set point of 1. and 10. (Table 5.4). The same values of the control parameters for the predictor controller were used in each case, but the PI controller was tuned for each case. The optimal control parameters are also shown in Table 5.4. Even when the PI controller was optimally tuned the preditor controller always showed better control. Figure 5.6 shows the response of the predictor controller to the noisy system as well as the change in the manipulated variable. Figure 5.7 shows the response for a step change of 10 in the disturbance. Using the predictor controller a step change of 1 in disturbance was not visible in the output. Figure 5.8 shows the response of the predictor controller to a step change of 1 in the set point. For this particular system the control using the predictor controller was approximately twice as good as what could be achieved with an optimally tuned PI controller. For systems with dead time the predictor controller would have shown even greater improvement.

Application of Preditor Controller To Chemical Reactor

In the previous example a system with known parameters was used to investigate the controller itself. In this section a considerably more complex example is considered to demonstrate the complete problem of determining the model parameters and then controlling the system. Thus, consider the stirred tank reactor shown in Figure 5.9 (7). The liquid feed to the reactor is at temperature $T_r\, (^\circ F)$, flow rate $w\,(lbs/min)$, and concentration $C_{ao}\, (lb\text{-}moles/cu\, ft)$ of reactant A. The product stream is of concentration $C_a\, (lb\, moles/cu\, ft)$, flow rate $w\,(lb/hr)$, and temperature $T_p\, (^\circ F)$. Since the contents of the tank.
<table>
<thead>
<tr>
<th>Step Change in Disturbance</th>
<th>Predictor $\Sigma e^2$</th>
<th>PI $\Sigma e^2$</th>
<th>K</th>
<th>K/Te</th>
<th>%Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.724</td>
<td>4.11</td>
<td>9.73</td>
<td>.37x10^{-8}</td>
<td>58%</td>
</tr>
<tr>
<td>1</td>
<td>1.787</td>
<td>4.58</td>
<td>10.71</td>
<td>.246</td>
<td>61%</td>
</tr>
<tr>
<td>10</td>
<td>6.181</td>
<td>7.01</td>
<td>30.63</td>
<td>1.87</td>
<td>12%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step Change in Set Point</th>
<th>Predictor $\Sigma e^2$</th>
<th>PI $\Sigma e^2$</th>
<th>K</th>
<th>K/Te</th>
<th>%Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.723</td>
<td>5.54</td>
<td>10.54</td>
<td>.152</td>
<td>51%</td>
</tr>
<tr>
<td>10</td>
<td>86.31</td>
<td>914.43</td>
<td>8.4</td>
<td>.35</td>
<td>90%</td>
</tr>
</tbody>
</table>
Figure 5.6. Predictive Control of Second Order System (Noise RMS = .2).
Figure 5.7. Predictive Control of Second Order System for a Step Change in Disturbance of 10
(Noise RMS = .2)
Figure 5.8. Predictive Control of Second Order System for a Step Change in Set Point of 1.
(Noise RMS = .2)
Figure 5.9. Predictive Control of Chemical Reactor.

INITIAL CONDITIONS:

- $V_t = 100 \text{ gal.} = 13.38 \text{ ft}^3$
- $V_B = 8.64 \text{ ft}^3$
- $A_B = 200.0 \text{ ft}^2$
- $\rho_t = 55.0$
- $C_{ao} = 0.9 \text{ Btu/lb °F}$
- $c_p = 1.0 \text{ Btu/lb °F}$
- $p_j = 62.4 \text{ lbs/ft}^3$
- $\Delta H = -12,000 \text{ Btu/lb-mole}$
- $U = 75.0 \text{ Btu/hr-°F-ft}^2$
- $T_p = 175°F$
- $T_k = 205.31°F$
- $m = 26.89 \text{ lbs/min}$
- $w = 73.5 \text{ lbs/min}$
- $T_{wi} = 80°F$
- $T_W = 218.86 °F$
are well mixed, the outlet concentration of A is approximately equal to the concentration of A in the tank.

In the reactor component A reacts to yield B according to the following irreversible reaction

$$2A \rightarrow B$$

This reaction is exothermic, with a heat of reaction $\Delta H$ of -12,000 BTU/lb mole. The rate of disappearance of A is given by

$$r_a = k C_a^2$$

where $k$ is the reaction rate constant. Although $k$ is independent of $C_a$, its dependence upon temperature is given by the following equation

$$k = k_c \exp\left(-k_a/T_p\right)$$

For this reaction, $k_c = 8.33 \times 10^8$ cu ft/lb-mole min and $k_a = 14,000$ °R. There is no volume change during the reaction.

To remove heat from the reaction mass, coolant is admitted to the tube bundle at temperature $T_{w1}$ (°F) and flow rate $m$ (lbs/min). The contents of the tube bundle are recirculated sufficiently rapidly to insure that the tube bundle temperature is uniform. In this example, the heat loss to the surroundings is neglected, as well as the heat generated by the mixer. The level in the reactor is assumed constant, as well as the heat capacity and density of all streams.

To mathematically describe this system without the controller, unsteady state enthalpy balances were written for the tube bundle and the kettle, as well as an unsteady state mass balance for component A in the kettle. For the kettle an unsteady state enthalpy balance reveals
An unsteady state mass balance for component A in the kettle indicates

\[ w C (T_R - T_p) + (-\Delta t) k C_2 v_T = U A_p (T_p - T_w) = v_T^p C_a (\frac{dT_p}{dt}) \]  

Equation 5.6

For the tube bundle an unsteady state enthalpy balance yields

\[ m C (T_{wi} - T_w) + U A_B (T_p - T_w) = v_B^p C_B (\frac{dT_w}{dt}) \]  

Equation 5.8

The above three equations, along with Equation 5.4, describe the reactor system without the controller.

To control the temperature of the reacting mass, a predictor controller will be used. The temperature of the product leaving the reactor will be measured and fed back to the controller which will manipulate the coolant flow rate \( m \) to the tube bundle.

All streams entering the reactor are noisy. There is variation in the mass flow rate of the reactants \( w \) (lb/min), in the temperature of the reactants \( T_R \) (°F), in the concentration of the reactants \( C_{ao} \) (lb-mole/ft³) and in the temperature of the coolant \( T_{wi} \) (°F).

The inlet streams were generated by superimposing noise on the steady state values. The noise was generated by passing white noise through a first order lag and adjusting the gain and the time constant to give the desired result. Three different noise levels were investigated and will be denoted as low, medium and high noise. The RMS value and the time constant for each inlet stream are shown in Table 5.5 as well as the resulting RMS of the temperature of the exiting stream. Figure 5.10 illustrated the feed stream fluctuations for the medium noise level over a 16 hour period.

The initial sampling interval of five minutes was chosen as one half the space time of the reactor. The pulse height of 10 lb/min of
TABLE 5.5

NOISE ON REACTOR FEEDS

<table>
<thead>
<tr>
<th>Mass Flow Rate of Reactants</th>
<th>Temperature of Reactants</th>
<th>Concentration of Reactants</th>
<th>Temperature of Coolant</th>
<th>Temperature of Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>RMS 2.5 50 T_f 50</td>
<td>RMS 5. 100 T_f 50</td>
<td>RMS .02 RMS 50 T_f 50</td>
<td>RMS .3 RMS 1. RMS 1.785</td>
</tr>
<tr>
<td>Med. 10.0 50 T_f 50</td>
<td>20. 100 T_f 50</td>
<td>.2 50 T_f 50</td>
<td>1.0 1. RMS 1.2 12.785</td>
<td></td>
</tr>
<tr>
<td>High 25.0 50 T_f 50</td>
<td>50. 100 T_f 50</td>
<td>.2 50 T_f 50</td>
<td>3.0 1. RMS 1.8 18.25</td>
<td></td>
</tr>
</tbody>
</table>
Figure 5.10. Feed Stream Fluctuations to the Chemical Reactor.
coolant was experimentally determined by increasing the pulse
height until a sufficient change was noticed in the outlet tempera­
ture. A random pulse was used as the forcing function due to the
work discussed in Chapter IV. The initial number of data points was
arbitrarily chosen to be 200. A rough check on the experimental
conditions can be made if the RMS value of the uncontrolled output
is available directly, or can be reconstructed from the model. A
rough estimate of the RMS value can be obtained very quickly from
the uncontrolled data as suggested by Goff (6).

\[
\text{RMS} = \frac{\text{Peak-To-Peak Value}}{\sqrt{2}}
\]

Using the pulse height of 10 lb/min and the experimentally determined
gain of 2.1 the effective pulse height at the outlet is 21°F. The
ratio of the RMS value of the product temperature to the effective
pulse height can then be used in Figure 4.5 to determine the necessary
number of data points. For this particular system 200 data points
were sufficient for both the low and medium noise, but 300 to 400
should have been used at the high noise level. This check gives a
rough estimate of the reliability of the experimentally determined
parameters. If it is found that the correct experimental conditions
were not used then additional data should be collected.

The results of using Pattern Search to determine the system
parameters are shown in Table 5.6. One run was made without noise
to determine the dynamic parameters of this system. A perfect fit
was not obtained in this case as the process is not a true second
order system. The fact that \( \delta_2 \) is negative prevents the calculation
of the time constants of the system. Even though no physical
<table>
<thead>
<tr>
<th>Noise</th>
<th>Run</th>
<th>Forcing Function</th>
<th>Sampling Interval min</th>
<th>λ</th>
<th>δ₁</th>
<th>δ₂</th>
<th>g</th>
<th>( \gamma_0 )</th>
<th>( \gamma_1 )</th>
<th>Criterion Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>1</td>
<td>Pulse of 10</td>
<td>5</td>
<td>.4636</td>
<td>.4604</td>
<td>-.1938</td>
<td>-2.028</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>2</td>
<td>Random Pulse of 10</td>
<td>5</td>
<td>.2375</td>
<td>.8064</td>
<td>.0406</td>
<td>-2.385</td>
<td>1.516</td>
<td>0</td>
<td>27</td>
</tr>
<tr>
<td>Medium</td>
<td>3</td>
<td>Random Pulse of 10</td>
<td>5</td>
<td>.1729</td>
<td>.8389</td>
<td>0</td>
<td>-2.884</td>
<td>1.754</td>
<td>0</td>
<td>765</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Random Pulse of 10</td>
<td>5</td>
<td>.1093</td>
<td>.9748</td>
<td>.1539</td>
<td>-2.428</td>
<td>1.300</td>
<td>.807</td>
<td>727</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Random Pulse of 10</td>
<td>2.5</td>
<td>.4370</td>
<td>.8732</td>
<td>.0247</td>
<td>-1.678</td>
<td>1.842</td>
<td>0</td>
<td>196</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Random Pulse of 10</td>
<td>2.5</td>
<td>.4313</td>
<td>.8523</td>
<td>-.0047</td>
<td>-1.713</td>
<td>.906</td>
<td>1.57</td>
<td>88</td>
</tr>
<tr>
<td>High</td>
<td>7</td>
<td>Random Pulse of 10</td>
<td>5</td>
<td>.9044</td>
<td>.8000</td>
<td>0</td>
<td>-2.904</td>
<td>1.999</td>
<td>0</td>
<td>1807</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Random Pulse of 10</td>
<td>5</td>
<td>.2147</td>
<td>.9412</td>
<td>.0425</td>
<td>-4.917</td>
<td>1.238</td>
<td>.831</td>
<td>1621</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>Random Pulse of 10</td>
<td>2.5</td>
<td>.5006</td>
<td>.4504</td>
<td>-.399</td>
<td>-1.825</td>
<td>.994</td>
<td>1.320</td>
<td>217</td>
</tr>
</tbody>
</table>
significance can be attached to $\delta_1$ and $\delta_2$, the system is still stable. At the low noise level (Run 2) a good fit of the experimental data was obtained ($\Sigma\alpha_p^2 = 27.5$). For the medium noise at a sampling interval of five minutes the criterion function increased to over 700. It was found that two minimums could be obtained, one with $\gamma_1 = 0$ and one with $\gamma_o = .8075$ (Runs 3 and 4). At these two minimums the dynamic parameters are almost identical but the stochastic parameters are greatly different. However, the difference in the fit is almost insignificant. The same results were also noted for the high noise at this sampling rate (Runs 7 and 8). By doubling the sampling rate a better fit of the experimental data could be obtained. Only one minimum was obtained for both medium and high noise and that was with $\gamma_1 \neq 0$. (Runs 6 and 9). The data for Run 5 ($\gamma_1=0$) was obtained by restricting the solution to illustrate the increased effect of $\gamma_1$ at this sampling rate. Thus, it has been demonstrated that a fairly good representation of this chemical reactor can be obtained with this model if the correct sampling rate is used.

Using the information obtained by Pattern Search for the various levels of noise the control of the chemical reactor was investigated. The limits on the flow rate of coolant were set at 0 and 200 lb/min to correspond to the valve being completely closed or open. Each run (Table 5.7) consisted of controlling the reactor for a period of 16 2/3 hours.

At the low noise level the controller reduced the standard deviation of the product temperature from 1.785 to .312 (Run 1). By tuning this controller the standard deviation could be reduced
<table>
<thead>
<tr>
<th>Noise</th>
<th>Tuned</th>
<th>Run</th>
<th>Sampling Interval</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$A_4$</th>
<th>$Y_0$</th>
<th>$Y_1$</th>
<th>$\Sigma e^2$</th>
<th>RMS</th>
<th>$\Sigma m^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>No</td>
<td>1</td>
<td>5</td>
<td>.311</td>
<td>-2.347</td>
<td>-1.893</td>
<td>-.095</td>
<td>1.516</td>
<td>0</td>
<td>19.486</td>
<td>.31</td>
<td>261.8</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>2</td>
<td>5</td>
<td>.312</td>
<td>-3.147</td>
<td>-1.593</td>
<td>-.095</td>
<td>1.516</td>
<td>0</td>
<td>13.679</td>
<td>.26</td>
<td>561.2</td>
</tr>
<tr>
<td>Medium</td>
<td>No</td>
<td>3</td>
<td>5</td>
<td>.209</td>
<td>-2.603</td>
<td>-2.183</td>
<td>0</td>
<td>1.755</td>
<td>0</td>
<td>726.550</td>
<td>1.91</td>
<td>29547.0</td>
</tr>
<tr>
<td></td>
<td>Yes</td>
<td>4</td>
<td>5</td>
<td>.165</td>
<td>-2.290</td>
<td>-2.510</td>
<td>-.19</td>
<td>1.753</td>
<td>0</td>
<td>650.300</td>
<td>1.81</td>
<td>30307.0</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>5</td>
<td>5</td>
<td>.121</td>
<td>-2.530</td>
<td>-2.460</td>
<td>-.389</td>
<td>1.300</td>
<td>.808</td>
<td>756.000</td>
<td>1.94</td>
<td>29387.0</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>6</td>
<td>2.5</td>
<td>.7584</td>
<td>-7.178</td>
<td>-6.117</td>
<td>.034</td>
<td>1.853</td>
<td>0</td>
<td>Uncontrolable</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>7</td>
<td>2.5</td>
<td>.7726</td>
<td>-6.095</td>
<td>-6.095</td>
<td>-.172</td>
<td>.907</td>
<td>1.577</td>
<td>515.450</td>
<td>1.60</td>
<td>671110.0</td>
</tr>
<tr>
<td>High</td>
<td>No</td>
<td>8</td>
<td>5</td>
<td>9.460</td>
<td>-18.005</td>
<td>-14.404</td>
<td>0</td>
<td>1.999</td>
<td>0</td>
<td>Uncontrolable</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>9</td>
<td>5</td>
<td>.273</td>
<td>-2.556</td>
<td>-2.406</td>
<td>-.109</td>
<td>1.238</td>
<td>.831</td>
<td>1796.700</td>
<td>3.00</td>
<td>105370.0</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td>10</td>
<td>2.5</td>
<td>1.003</td>
<td>-7.286</td>
<td>-3.281</td>
<td>2.907</td>
<td>.999</td>
<td>1.321</td>
<td>Uncontrolable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
to .261 (Run 2), but the amount of valve motion was increased. Better control in this example was always found to correspond to increased valve motion.

At the medium noise level the controller reduced the standard deviation of the product temperature from 12.06 to below 2.0. The additional noise also increased the valve motion by about a factor of 10. Runs 3 and 4 show the effect of tuning the controller at this noise level. Run 5 at a sampling interval of five minutes indicated that the inclusion of $\gamma_1$ did not help control. However, when the sampling rate was doubled the inclusion of $\gamma_1$ was necessary to control the system. Run 7 should have shown a $\Sigma e^2$ of less than 100, but due to the limitations of the valve this was unattainable. During this run quite frequently the valve was completely closed or open as shown by the large $\Sigma m^2$, but the overall control was the best obtained at this noise level. In Run 6 where $\gamma_1=0$ no control was obtained as the valve was always completely closed or open.

At the high noise level the controller reduced the standard deviation of the temperature of the product from 18.25 to 3.0. Again the valve motion was increased by a factor of 4 over that required by the medium noise. The $\gamma_1$ parameter was found necessary for control to be maintained at all (Run 9). In Run 8 at the slow sampling rate and in Run 10 at the fast sampling rate no control was obtained as the valve was either open or closed at all times. For Run 10 if control would have been possible the $\Sigma e^2$ would have been near 200.

From this example it is apparent that the quality of control predicted by the fit of the experimental data is not always attainable. The better the control the more valve movement is required and
thus the quality of control is limited by the range of the manipulated variable. For this particular example the control at the medium and high noise levels could have been improved by altering the physical system. Increasing the system gain by increasing the area of the tube bundle would have made better control possible.

A comparison was made at the medium noise level of an optimally tuned PI controller and the predictor controller used in Run 7. The results of a 16 hour run are shown in Figure 5.11 and Table 5.8. For this particular case the PI controller did almost as well as the predictor controller showing only a 12% improvement untuned and 21% improvement tuned.

Summary

A control strategy was designed based on the model developed in Chapters III and IV. The stochastic portion of the model was used to predict the error from set point a number of sampling periods in advance. The dynamic portion of the model is then used to calculate the value of the manipulated variable that will counteract this predicted error. Applying this control strategy to a second order system including dead time showed that the change in the manipulated variable could be reduced by adjusting $g$, $\lambda$ or $\nu_0$ without adversely affecting the control error. This particular system became uncontrollable for a dead time greater than .5. Decreasing the sampling rate was found to be the best solution to this problem. The predictor controller showed on the average a 50% improvement over an optimally tuned PI controller for a noisy system and step changes in both disturbance and set point.
Figure 5.11. Comparison of Predictive Control and PI Control of a Chemical Reactor.
**TABLE 5.8**

**COMPARISON OF PI AND PREDICTOR CONTROL OF A CHEMICAL REACTOR**

<table>
<thead>
<tr>
<th>Predictor Controller</th>
<th>PI Controller</th>
<th>$K$</th>
<th>$K/T_s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untuned</td>
<td>Tuned</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\Sigma e^2$</td>
<td>726.55</td>
<td>650.30</td>
<td>826.98</td>
</tr>
<tr>
<td>$\Sigma n^2$</td>
<td>29,547</td>
<td>30,307</td>
<td>22,932</td>
</tr>
</tbody>
</table>
The complete procedure for obtaining the model and controlling the system was then demonstrated on a chemical reactor. It was found that although a very good fit of the experimental data could be obtained, that equivalent control was not always possible. Since improved control also means more change in the manipulated variable the amount of improvement is limited.


NOMENCLATURE

\[ Z_p \] = magnitude of disturbance at time \( p \)

\[ Z_p(t) \] = best predictor of \( Z_{p+1} \) made at time \( p \)

\[ \alpha_p \] = uncorrelated random variables or white noise

\[ S \] = summation operator

\[ V \] = backward difference operator

\[ \gamma_j \] = parameters in stochastic model

\[ x_p \] = level of manipulated variable at time \( p \)

\[ m_p \] = change in manipulated variable between time \( p-1 \) and \( p \)

\[ X_p \] = accumulated compensation at output

\[ x_p \] = compensation between time \( p-1 \) and \( p \)

\[ z \] = steady-state gain of system

\[ \delta_j \] = parameters in dynamic model

\[ \delta_j \] = parameters in dynamic model (function of \( \delta_j \))

\[ \lambda \] = dead time parameter

\[ R_p \] = setpoint at time \( p \)
APPENDIX A

REVIEW OF OPTIMIZATION TECHNIQUES
APPENDIX A

Optimum Gradient

The method of optimum gradient employed is due to Beckey (1). The steps are:

1. Determine the direction of the gradient at the starting point $P_0$.
2. Perform a linear search in the negative gradient direction to locate minimum point $P_1$.
3. Repeat steps 1 and 2 until minimum is located.

Discrete approximations for the partial derivatives are used in the calculation of the gradient and are obtained by perturbing each parameter by .01% as suggested elsewhere (17). The method is called optimum gradient as an optimum point is located in each linear search and an optimum step size is calculated based on a modified Newton-Raphson method. The step is taken in the negative gradient direction and the criterion function is evaluated at this point and halfway between this point and the original point. On the basis of these values the step size is doubled or halved until the approximate minimum is located. In order to determine the minimum more accurately, quadratic interpolation is used over the last three values of the criterion function. The step size for future iterations is determined from the results of previous iterations. The method is terminated when the gradients and parameter changes become very small.
Advantages are:

1. For starting values relatively far from the optimum, this method will usually pick good directions and reduce the criterion function rapidly during the first few iterations.
2. The method will inherently stay away from saddlepoints (21).

Disadvantages are:

1. Since the analytical partial derivative functions are usually not available or are impractical to evaluate, numerical methods were used to estimate the partials with the resulting inaccuracies.
2. Very slow convergence is incurred in most cases as the optimum is approached. Theoretically it will take an infinite number of steps to reach the optimum.
3. The method has a tendency to oscillate.

**Gradient Partan**

Gradient Partan or the method of parallel tangents stems from the fact that, for a two dimensional positive definite quadratic criterion function, the line joining the base point and the point obtained after two iterations of optimum gradient passes directly through the optimum. The steps in this method implemented are:

1. To start the search perform two iterations using the technique of optimum gradient from point $P_1$ to locate points $P_2$ and $P_3$ (refer to Figure A-3).
2. Perform a linear search along the line connecting points $P_1$ and $P_3$ to locate minimum $P_4$.
3. Determine the direction of the gradient at the point $P_4$. 
Figure A-3. Gradient Partition.
4. Perform a linear search in the negative gradient direction to locate minimum \( P_5 \).
5. Perform a linear search along the line connecting points \( P_2 \) and \( P_5 \) to locate minimum \( P_6 \).
6. Continue with one step in the negative gradient direction and then one step in the acceleration direction until the minimum is located.

The computer program for this method uses the linear search technique present in the optimum gradient program. The optimum step size is calculated separately for the acceleration steps and the gradient steps. For a quasi-quadratic function the method of gradient partan will converge to the optimum in \( 2N-1 \) unidimensional searches provided no error is incurred during gradient calculations or linear searches. For functions that are not quasi-quadratic, partan will not reach the optimum in \( 2N-1 \) steps. It is also best not to locate the minimum for the one dimensional search too accurately as this may cause sticking on a resolution ridge (21). The method is terminated when the parameter change and the gradients become very small.

Advantages are:

1. As the first few iterations are in the gradient direction, bad starting points will not be a handicap.
2. The acceleration step removes some of the oscillation present in optimum gradient and gives the method the ability to follow ridges.

Davidon's Method

The variable metric method of minimization developed by William C. Davidon is also a gradient method (3). In the process of locating
each minimum a matrix which characterizes the behavior of the function about the minimum is determined. For a region in which the function depends quadratically on the variables, no more than $N$ iterations are required, where $N$ is the number of variables.

Outline of Method

1. The metric matrix and the gradient are used to establish a search direction.

2. Successively larger steps are taken in this direction until the relative minimum is bracketed.

3. A linear search is performed to locate the minimum within the interval. The improvement in the criterion function is compared with a step perpendicular to this direction at the relative minimum.

4. The metric matrix is updated based on information about the function obtained in this direction.

In the neighborhood of any point the second derivatives of $f(x)$ specify a linear mapping of changes in position, $dx$, onto changes in gradient $dg$. For example in the case of the $i^{th}$ variable

$$
\frac{d^2 f}{dx_i dx_j} = \sum_{j=1}^{N} \frac{\partial^2 f}{\partial x_i \partial x_j} \, dx_j
$$

where \( \sum_{i=1}^{N} \frac{\partial^2 f}{\partial x_i \partial x_i} = H_{ij} \) or the Hessian matrix. If this matrix were constant and explicitly known, then the value of the gradient at any point would suffice to determine the minimum. In this case the desired step $\Delta x_j$ would be given by

$$
\Delta x_j = \left( H_{jj} \right)^{-1} \frac{\partial f}{\partial x_j}
$$
Since the matrix $H_{ij}$ is neither constant nor known this method employs an iterative technique to improve the estimates of $H_{ij}$ based on the changes in the gradients and the parameters. To start the method, $H_{ij}$ is set equal to the unity matrix, making the first step in the direction of steepest descent. After every $N+1$ iterations the metric matrix is set back equal to the unit matrix. In the linear search portion of the program cubical interpolation is used to locate the minimum. The method is designed for problems where an analytical expression is available for the derivatives. However a modification of this method has been published by Stewart (19) which indicates how to use difference approximations for the derivatives.

Advantages:

1. This method is the basis for one of the most powerful gradient methods known to date.
2. The method has the property of quadratic convergence.
3. As the dimensionality of the search increases the number of function evaluations only increases in a linear manner.

Disadvantages:

1. The method works best when analytical expressions are available for the derivatives.

Conjugate Gradients

The method of conjugate gradients used in this survey was developed by Fletcher and Reeves (17).

Outline of Method

1. At the starting point $x_0$, evaluate the gradient $g_0$ and set the search direction $d_0 = -g_0$. 
2. Perform a linear search in the direction \( d_i \) on the line through \( x_i \) to locate the relative minimum \( x_{i+1} \).

3. Evaluate the gradient \( g_{i+1} \) at \( x_{i+1} \) and calculate

\[
\tilde{\eta}_i = \frac{g_{i+1}}{g_i^2}
\]

4. Calculate the new direction \( d_{i+1} = -g_{i+1} + \tilde{\eta}_i d_i \).

5. The method is terminated if \( g_i = 0 \) or \( N + 1 \) iterations starting from a steepest descent search, produces no reduction in the value of the function, otherwise a new iteration is started.

This process is guaranteed, apart from rounding errors, to locate the minimum of any quadratic function of \( N \) arguments in at most \( N \) iterations. For functions which are non-quadratic the process is iterative. After every \( N + 1 \) iterations a step is taken in the negative gradient direction to prevent oscillation and slow convergence.

The linear search used in this program is the same one used in both Davidon's method and the Fletcher-Powell modification.

Advantages:

1. Particular advantages of this method are its simplicity and its modest demands on storage space (only three vectors are required).

2. This method also exhibits quadratic convergence.

**Fletcher, Powell Modification of Davidon's Method (11)**

This method is a modification of the original method devised by W. C. Davidon (8) and published in 1959.

**Outline of Method**

1. Given the parameter vector \( x_i \) and the gradient vector \( g_i \), calculate the direction \( d_i \):
\[ d_i = -H_i s_i \]

where \( H_i \) is the metric matrix which is initially the unity matrix.

2. Find \( \sigma_i \) so that the criterion function \( f(x_i + \sigma_i d_i) \) is a minimum along \( x_i + \sigma_i d_i \).

3. Set \( x_{i+1} = x_i + \sigma_i d_i \) and calculate \( g_{i+1} = g(x_{i+1}) \) and \( y_i = \frac{g_{i+1} - g_i}{\sigma_i} \).

4. Form \( H_{i+1} \) by

\[
H_{i+1} = H_i + \sigma_i \frac{d_i d_i^T}{y_i} - \frac{H_i y_i y_i^T H_i}{y_i^T H_i y_i}
\]

5. Terminate the search when the direction \( d_i \) and the parameter \( \sigma_i \) become less than some minimum.

This method like Davidon's has quadratic convergence and will require \( N \) iterations to converge for a quadratic function. If convergence in general is not reached in \( N \) iterations or if the directional derivatives are positive the metric matrix is set equal to the unity matrix and the search started again. This method like Davidon's was designed to use analytical expressions for the derivatives. If difference approximations are used care must be taken to obtain accurate values. This may mean going to central differences in certain cases. The advantages and disadvantages are basically the same as for Davidon's.

Marquardt's Method

This method developed by Marquardt (14) provides for an automatic transition from the method of steepest descent to Gauss-Newton as the minimum of a function is approached.
Outline of Method

1. Form the matrix $A$ and $E$ where

$$b^{\times N} = \frac{\partial f_i}{\partial b_j} \quad i = 1, 2 \ldots M \quad j = 1, 2 \ldots N$$

$$A^{\times N} = b^T$$

$$E = b^T(R-f_o) \quad R = \text{experimental data}$$

$M$ = number of data points

$N$ = number of parameters

$f_i$ = value from mathematical model at point $i$

$X$ = parameters

2. Scale the matrix $A$ and the vector $b$ by the square root of the diagonal elements of $A$.

3. Solve

$$(A^* + \lambda I)\zeta^* = b^* \quad \zeta_j = \frac{\zeta_j^*}{\sqrt{A_{jj}}}$$

4. Evaluate the criterion function at $X^{t+1} = X^t + \zeta$

If the criterion function is less than the previous go to step 1. If the criterion function is greater than the previous adjust the value of $\lambda$ and go to step 3.

A sufficiently large $\lambda$ always exists which will give a $\zeta$ value that will decrease the value of the criterion function if $X$ is not at the minimum. When $\lambda = 0$ this method is identical to the Gauss-Newton method and when $\lambda \to \infty$ it is similar to steepest descent. Intermediate values of $\lambda$ give trajectories between that of steepest descent and Gauss-Newton. Several values of $\lambda$ will usually be tried during each iteration to find the best, as it takes considerable effort to
reevaluate A and b. This method tries to utilize the best qualities of two different techniques, steepest descent for points far removed from the minimum and Gauss-Newton in the vicinity of the minimum.

**Rotational Discrimination**

The method of rotational discrimination, developed by Law and Farris (9), is probably one of the most sophisticated methods available.

**Outline of Method**

1. Select a base point, \( p^0 \), and a maximum allowable distance factor \( L \).
2. Compute \( \frac{\partial f(x^0)}{\partial x} \) and \( \frac{\partial^2 f(x^0)}{\partial x^2} \).
3. Find the eigenvalues and eigenvectors of \( \frac{\partial^2 f(x^0)}{\partial x^2} \) and calculate \( C = S^T \frac{\partial^2 f(x^0)}{\partial x^2} S \) ordering the diagonal elements in descending algebraic order. \( S \) is a vector of scale factor.
4. Starting with \( y_1 \), the elements of \( y \) are computed by
   \[
   y_i = -\frac{\partial f}{\partial y_i} / C_{ii}
   \]
   until
   a) \( i = N \)
   b) \( C_{ii} \leq 0 \)
   c) \( y_i > L \)
5. When the sequence is terminated for reasons (b) or (c) above at the \( k \)th parameter, then a switch is made from Newton's method to steepest descent logic. For the \( k \)th parameter, \( y_k \) is then assigned
a scale factor of 1 and

\[ y_k = -L \text{ sgn} \left( \frac{\partial f}{\partial y_k} \right) \]

6. If there are parameters in the list after the \( k \)th one, the steepest descent logic is continued until the end of the parameter list is reached.

\[ y_i = y_k s_i \frac{\partial f}{\partial y_i} / \frac{\partial f}{\partial y_k} \]

7. The resulting \( Y \) vector is converted back into a \( \Delta x \) vector by \( \Delta x = SY \) and a one-dimensional search along the vector \( x = x^0 + \Delta x \) is performed.

8. The distance factor is updated from

\[ \frac{L_{i+1}}{L_i} = \exp \left[ 0.88255 \tan^{-1} (0.56654 \ln \sigma) \right] \]

9. Convergence is achieved when either

a) the change in \( f \) between iterations is within a tolerance.

b) \( \sigma \Delta X \) becomes very small.

c) each \( \frac{\partial f}{\partial x_i} \) becomes very small.

If convergence is not met, go to step 2.

Rotational discrimination is basically a tactic for choosing a superior search vector in terms of rotated coordinates which exhibit zero local interaction. The program requires external scale factors so that a move in any direction will cause approximately the same change in the criterion function. Based on the assumption that this can be done rotational discrimination attempts to combine the best features of Newton's method and a form of steepest descent to compute
a search vector. If reasonable external scaling cannot be provided, the algorithm will still operate, but will tend to resemble either Gauss-Newton or steepest descent in its behavior, usually the former.

Pattern Search

The pattern search presented in this paper is discussed on page 307 of Reference (21).

Outline of Method

1. Evaluate the criterion function at the initial point $b_1$.
2. Make a trial step in each of the coordinate directions. If the value is better than the previous point use it as a base point and step about it in the other directions. If the criterion function is higher than the best value to date step in the opposite direction and evaluate the criterion function. If no improvement is found for this coordinate direction, step in another direction from the best point available.
3. After each coordinate direction has been tried an acceleration step is made in the direction of the line connecting the original and final comparison points used in each iteration in step 2. The length of the acceleration step is the distance between these two points. The criterion function is not evaluated at this point but step 2 is initiated using the best point available for comparison.
4. The method is terminated when a step in all directions yields no improvement.

To increase the efficiency of the computer program a large step-size is used initially and after it fails to yield improvement the step
size is reduced by a factor of ten. The program terminates after a specified number of step reductions have been made. The successful directions are stored during each iteration and are tried first in the next iteration.

Advantages:

1. Derivatives do not have to be evaluated.
2. As the number of variables increases the computation time increases only as the first power of the number of variables.
3. The method has the ability to follow a ridge.
4. Simple inequality constraints can be incorporated into the method.

Disadvantages:

1. An initial step size must be read into the program. The value of this step size effects the efficiency of the search.

Powell's Method

This method minimizes a sum of the squares of non-linear functions by changing one variable at a time. Each iteration starts with a search along \( N \) linearly independent directions starting from the best known approximation to the minimum. Initially the directions are chosen as the coordinate directions.

Outline of Method

1. For a base point \( x_0 \) for \( i = 1, 2 \ldots N \), where \( N \) is the number of variables, calculate \( \sigma_i \) so that the criterion function \( f(x_{i-1} + \sigma_i d_i) \) is a minimum and define \( x_i = x_{i-1} + \sigma_i d_i \).
2. Find the integer \( k, (1 \leq k \leq N) \) so that \( \{f(x_{k-1} + f(x_k)) \}

is a maximum and define \( \Delta = f(x_{k-1} - f(x_k)) \).
3. Calculate \( f_3 = f(2x_N - x_0) \) and define
\[
\begin{align*}
  f_1 &= f(x_0) \\
  f_2 &= f(x_N)
\end{align*}
\]

4. If either \( f_3 \geq f_1 \) and/or
\[
(\tilde{d}_1 - 2f_2 + 3) \cdot (f_1 - \tilde{d}_2 - \Delta)^2 \geq \frac{1}{3} \Delta (f_1 - f_3)^2
\]
then use the old directions \( d_1, d_2 \ldots d_n \) for the next iteration and use \( x_n \) for the next \( x_0 \). Otherwise,

5. Define \( d = x_N - x_0 \), calculate \( \sigma \) so that \( f(x_N - \sigma d) \) is minimum, and use \( d_1, d_2 \ldots d_{m-1}, \tilde{d}_{m+1} \ldots d_n \) as the directions and \( x_n + \sigma d \) as the starting point for the next iteration.

The linear search in this method is performed by fitting a quadratic equation to three function evaluations in order to predict the turning point of the quadratic. If this point is within the maximum step allowed the criterion function is evaluated there and replaces one of the original three points. If the predicted step exceeds the maximum step allowed, a step is taken equal to the maximum. This is repeated until two points fall within the required accuracy which is then designated the minimum.

An alternate technique of this method was also programmed. The same basic logic is used but a different linear search—an open ended Fibonacci search. Since the extreme points are not known the minimum must be bracketed by stepping out in an accelerated manner from the base point. Once the minimum is bracketed a Fibonacci search is performed to locate the minimum within the desired accuracy.

Both methods have been programmed so that initially the required accuracy of the linear search can be low and increased as the minimum...
is approached.

Advantages:

1. This method converges rapidly from a bad approximation.
2. Every variable has a chance of changing an equal number of times, whereas in Smith's method (18) this is not true.

Rosenbrock's Method

Rosenbrock's hill climbing method is a pattern search method for finding maximums or minimums and can be used when the variables are restricted to a certain region.

Outline of Method

1. Trial steps are taken in the search direction and the step size adjusted until one step has failed and one step has succeeded in each direction. All successful steps are used as new base points. Initially the coordinate directions are used as the search directions.

2. New directions are now chosen using the following technique.

Let \( c_1^o \) be the algebraic sum of all the successful steps in the direction \( d_1 \). Then let

\[
A_1 = c_1^o d_1 + c_2^o d_2 + \cdots + c_N^o d_N
\]

\[
A_2 = c_2^o d_2 + \cdots + c_N^o d_N
\]

\[
A_N = c_N^o d_N
\]

\[
B_N = \frac{1}{N-1} \sum_{j=1}^{N-1} A_j \cdot d_j^1 \cdot d_j^1
\]

\[
d_N^1 = B_N / |B_N|
\]

Return to step 1 using the new directions.
$A_1$ is the vector joining the initial and final points in step 1 and $A_2$ is the sum of all the advances made in directions other than the first. The $A$ vectors are then transformed into a set of orthogonal unit vectors $B$ and finally into the new direction vectors $d$. By using this method of determining the directions the vector, $d_1$ is always along the direction of the fastest advance and $d_2$ along the best direction normal to $d_1$ and so on. This program always starts out with an initial step size of .1. The step length thereafter is adjusted by a factor of .5 or 3, as the logic requires.
LITERATURE CITED


APPENDIX B

QUASILINEARIZATION COMPUTER PROGRAMS
QUASILINEARIZATION

GENERAL PROGRAM
COAST LINEARIZATION

STATE VARIABLES DEFINED IN TERMS OF THE PREVIOUS SOLUTION

C CURRENT X(23), BETTA(23), DT, TOL, ST, C1, C2, A(8), X(14, 15), A = CLAS 1
1APAT(14, 14), ETNI(14), CC(14), PEI, X(14, 15) = CLAS 2
C CURRENT NS, IPAT, N1, NORD, NORDP1, NORDP2, NCR, NCNP3, NCNP2, NCNP1, INTEGRAL CLAS 3

10C READ(5, 7) NSPC, NS, TI, TCPS, AT, IPRT
C NCRD = ORDER OF MODEL EQUATION
C NCS = NUMBER OF OBSERVATIONS
C CT = INTEGRATION DELTA T
C TCPS = TIME BETWEEN OBSERVATIONS
C MTE = MAXIMUM ALLOWABLE DEBT TIME
C IFMAT = PRINT CONTROL
C IPNT = ONLY THE ANSWER IS PRINTED AT EACH ITERATION
C IPNT = 1 MORE INFORMATION PRINTED AT EACH ITERATION
C ITR = ITERATION NUMBER
C PHI = VALUE OF SUM OF THE SCALES.

7 FORMAT(232, 2F12.3, 2E12.3)

IF(NCRD > 0, 0) CALL EXIT
WRITE(16, 8) NCRD, NSPC, CT, TCPS

80 FORMAT(27F12, 2E12, 3/26F12, 2E12) NUMBER OF OBSERVATIONS = 16/230 NUMBER OF OBSERVATIONS CLAS 3
115/125. TIME INCREASES 0.3/26. TIME BETWEEN OBSERVATIONS = 0.3 CLAS 13
NCRP1 = NCRD + 1
NCRP1 = NCRD - 1
NCRP2 = NCRD + 2
AOR = 2 * NCRD
AORP2 = NCRD + 2
AORP3 = NCRD + 3
CALL XIN
CALL ASSUME
2 CALL SOL(1)
CALL CAUS(NCRP2)
CALL SOL(2)
C
CHECK FOR CONVERGENCE
IF (ABS(A(IAR)+CC(I))-CC(NAR)) .GT. .001) GO TO 1

WRITE(6,3) ITR,PH
CC 16 1 = 1,ACRC
WRITE(6,4)I,A(I)
WRITE(6,23)A(NCRP1)
WRITE(6,21)A(NCRF2)
C
FORMAT(133) DEAD TIME = E14.4
GC TC 1 ,
C
1 CC 13 1 = 1,NCRP2
J = ACRC + 1
C
WRITE(6,3) ITR,PH
C
FORMAT(115): ITERATION 15/78 P11 = E15.5
GC 22 1 = 1,ACRC
C
WRITE(6,4)I,A(I)
C
FORMAT(123): A = E14.4
WRITE(6,23)A(NCRP1)
C
FORMAT(16) GAMA = E14.4
WRITE(6,21) A(NCRF2)
IF (ITR .LT. 25) GC TC 1 J
ITR = ITR + 1
GC TC 2
C
IF (ITR .LT. 25) GC TC 1 J
C
SUBROUTINE XIN
C
THIS SUBROUTINE GENERATES DATA FROM THE MODEL EQUATION.
COMMON A(24),BETA(24),DT,ICM(6,14),A(8),X(14,15),
IAMAT(14,14),RVECT(14),CC(14),PH,IA(14,15),IC
C DIMENSION AA(2),CA(1.),CR(1.)
C & I = 1,3
3 AA(I) = I.
READ(E,1.) NC,(AA(I),I = 1,3)
C & NC = ORDER OF DATA
C AA = PARAMETERS USED TO GENERATE DATA
C AA(I) THRU AA(8) = DESIRED PARAMETERS RELATED TO TIME CONSTANTS
C OF THE SYSTEM
C AA(9) = GAIN OF THE SYSTEM
C AA(10) = COMPARATIVE TIME OF SYSTEM IN SAMPLING Periods
10 FORMAT(1i5,1f5.2)
C THIS PROGRAM REQUIRES A RANDOM NUMBER GENERATOR
C AN1 = FACTOR TO ADJUST HEIGHT OF UNMEASURED INPUT NOISE
C ANC = FACTOR TO ADJUST HEIGHT OF UNMEASURED OUTPUT NOISE
C AN2 = FACTOR TO ADJUST HEIGHT OF MEASURED NOISE ON OUTLET
C N = NUMBER OF SIGNIFICANT FIGURES IN DATA
READ(E,9) AN1,ANC,AN2,N
9 FORMAT(13f1.5,1i1)
WRITE(6,12) AN1,ANC,AN2,N
12 FORMAT(12H1,INPUT NOISE,15.5,6X,12HOUTPUT NOISE,15.5/
12H,MEASURED INPUT NOISE,15.5)
X = 0.94
Y = 0.93
Z = RAND(X,Y)
XC = YC
ADP1 = NP - 1
AT = TCES/ST + .6
NCSP1 = NCRS + 1
NCSP2 = NCRS + NP + 1
DC 11 I = 1,NCSP1
C FORCING FUNCTION IS A DAMPED SINE
R(I) = (SIN(FLDA1(I))*0.32) & *10.*EXP(-FLDA1(I)*0.01)
JJ = 1**.**RS * R(I)
11 R(I) = FLCAT(JJ) / 1, *WS
   CC  YC  I = 1, NCSET
80 R(I) = R(I) + FREAD(XC,YC)*A(I)+2. - ANI+RAND(CX,YC)*AN*2.-ANN
   CC  8 I = 1, ND
   CB(I) = 
8 CA(I) = .
   TF = FA(I)
   J = TF
   FJ = J
   K = 1 - J
   IF(K) 15,19,26
19 FLAG1 = .
   CC TC 7:
20 FLAG1 = (FJ+1.-TF)*R(K)
70 CC  I = 1, NCSET
     BETA(I) = CA(I) + FADD(XC,YC)*AX2 = 2. - ANU
     JJ = 1, *WS = ELTA(I)
     BETA(I) = FLCAT(JJ) / 1, *WS
     FLAG = FLAG1
     K = I - J + 1
     IF(K) 22,22,24
24 IF(K-1) 26,26,28
28 FLAG1 = (FJ+1.-TF)*R(K) + (11-FJ)*R(K-1)
   CC TC 5.
22 FLAG1 = .
   CC TC 5:
26 FLAG1 = (FJ + 1. - TF) * R(K)
50 CR = (FLAG1 - FLAG)/FLCAT(HI)
   CC 1 L = 1, KT
   IF(KS, EC.1) CT(CT)
   CC 3 HI = 1, NCCT
   KK = HI + 1
5 CA(II) = CA(II) + CT * CB(KK)
93 CONTINUE
CC 4 II = 1, NC
CA(KC) = CA(NC) + II * (-MA(II)@CP(II))
CA(NC) = CA(NC) + (IA(5) + FLG) * DT
CC 51 II = 1, NC
CR(II) = C(A(II))
1 FLG = FLG + CF
CC 13 I = 1, NC
II
R(I) = (1.R(FLOAT(I)@.38) )
1RANV(XC,YC) = AAF = 2. - AAF
JJ = 10, *NS * F(I)
13 R(J) = FLG(J) / 1. *NS
WRITE(6,62) NC
60 FORMAT(1F:25X,4F14.10 ORDER LF DATA 15)
CC 61 I = 1, B
61 WRITE(6,62) I, AA(I)
62 FORMAT(2F:AI12,2F = F7.4)
WRITE(6,63) AA(5)
63 FORMAT(2F:CAIN = F7.4)
WRITE(6,64) AAIN
64 FORMAT(13H:DEAE TIME = F7.4)
WRITE(6,61) PHF(1), BETA(I), I = 1, NOAE
2 FORMAT(1X, 2F27.6)
RETURN
END

SUBROUTINE ASSUME
SUBROUTINE ASSUME GENERATES INITIAL SOLUTION
CCPC(R(.2,2),PLTA(.2,2),DT,T15S,CC 6,1,14),A(8),X(14,15),
1ASM(14,14),BVC(14,14),PI1,X(14,15),NC
CONCP(NCES,IFRILAT,NCER,NCER1,NCER2,NCER3,NCER4,NCER5,NCER6)
12 FORMAT(78F,PRI FOR ASSUMED SOLUTION = E15.5)
12 FORMAT(6F15.5)
L = 1.
PHI = 0.
REAC(5,12)(A(1),I = 1,ACRBP?)

C
A = ASSUMED PARAMETERS
C
THE PARAMETERS ARE READ IN CONSECUTIVE FIELDS WITH THE LAST
C
TWO BEING THE GAIN AND THE FLAG TIME RESPECTIVELY
C

CC 2 I = 1,ACRP

2 CC(I,J) = 0.
CR = .
ACRBP1 = NOFS + 1
TF=A(ACRBP)
J=TF
FJ = J
K = I - J
IF(K) 1S,19,21
19 FLAG1 = .
CC IC 7:
20 FLAG1 = (FJ + 1. - TF)*R(K)
78 CC 3 I = 1,ACRBP1
FLAG = FLAG1;
K = I - J + 1
IF(K) 22,22,24
24 IF(K-1) 26,26,28
28 FLAG1 = (FJ+1,-TF)*R(K) + (1-FJ)*R(K-1)
CC IC 5:
22 FLAG1 = .
CC IC 5:
26 FLAG1 = (FJ + 1. - TF)*R(K)
53 CR = (FLAG1 - FLAG)/FLOAT(HT)
CC 1 II = 1,ST
CC = C(ACRBP,L)
CC 4 II = 1,ACRC
4 CP = CR + DT = (-A(11) * C(I1,L))
C(ACRP+1) = CP + A(ACRP+1) * FLAG * DT
IF(ACRP < EQ. 1) CC IC 7
CC 8 II = 1,ACRP1
KK = 11 + 1
8 C(11, L + 1) = C(11, L) + BT*C(KK, L)
7 IF( L .NE. 1) GO TO 6
6 IF(IFPR1 .EQ. 1) WRITE (6, 5)
5 FORMAT(14: L, 5X, 2FC1, 8X, 2HC2, 8X, 2FC3, 8X, 2HC4, 8X, 2HC5, 8X, 2HC6, 8X,
12HC7, FX, 2FC8, 8X, 2FC9, 8X, 3FC1.)
4 IF(IFPR1 .EQ. 1) WRITE (6,9) L, (C(P, L), P=1, MCRD)
3 FCRPAT(15, 11, 12, 4)
2 FLAG = FLAG + CR
1 L = L + 1
0 IF(I .EQ. 1, MCRS) GO TO 2
IF(I .LT. MTH) GO TO 3
LL = ATF - (L - 1) + 1
PF1 = PH1 + (BETA(I) - C(1, LL))*E2
3 CONTINUE
WRITE (6, 1) PF1
RETURN
END

SUBROUTINE SOLNF(IFPR1, IK)
SUBROUTINE SOLNF GENERATES THE FUNDAMENTAL SOLUTIONS
C
C
IFPR1, K

1 IFPRAT (14, 14), BVEC(14), CC(14), PH1, XA(14, 15), IC
CCMFCB, ACBS, IFRAT, AT, MCRD, MCRP1, MCRP2, MCRP3, MCRP4, MCRP5, MCRP6, MTH
MT = MTH + 1
MCRP1 = MCR + 1
L = MTH * NT + 1
CC 2 J = 1, MCRP2
CC 1 J = 1, MCRP3
XA(I,J) = 0.
1 X(I,J) = 1.
XA(I,1) = 1.
2 X(I,1) = 1.
TH = A(MCRP2) + ET / TCRS
J = TF
FJ = J
KL = MT - J
IF (TF .LT. J) CC TC 2
FLAG2 = (FJ+1.-TF) * R(KL ) + (TF-FJ) * R(KL-1)
CC TC 21
23 FLAG2 = (-FJ+1.+TF)*R(KL) + (-TF+FJ)*R(KL+1)
21 TF = A(ACRIP2)
J = TF
FJ = J
KL = MT - J
IF (TF .LT. J) CC TC 22
FLAG = (FJ+1.-TF) * R(KL) + (TF-FJ) * R(KL-1)
CC TC 22
22 FLAG = (-FJ+1.+TF)*R(KL) + (-TF+FJ)*R(KL+1)
23 CC 8 I = MT ,ACES
IF (ICFT .NE. 1) CC TC 17
IF (IPRAY .NE. 1) CC TC 11
WRITE(*,61)
65 FORMAT(12H; OBSERVATION IS)
CC 50 K = 1,ACRF2
56 WRITE(6,61) (X(*,N),N = 1,ACRP3)
61 FORMAT(1X,7F14.5)
11 CALL MAT3X(1)
10 FLAG2 = FLAG
KL = I - J 41
IF (TF .LT. J) CC TC 24
FLAG = (FJ+1.-TF)*R(KL) + (TF-FJ)*R(KL-1)
CC TC 25
24 FLAG = (-FJ+1.+TF)*R(KL) + (-TF+FJ)*R(KL+1)
25 CC 3 K = 1,MT
IF (ICFT .EQ. 2) CALL PESEN(L)
CC 4 LL = 1,ACRP3
IF (I > P, FC, 1) CC TO 7
CC 5 NN = 1,ACRP
5 X(NP,LL) = X(NP,LL) + DT*XA(NP+1,LL)
CC 6 KK = 1,ACRP
II = ACHR + KK
6 X(NCRC,LL) = X(NCRC,LL) + CI*(1-A(KK)*XA(KK,LL)) + (-C(KK,L))
1 XA(II,LL))
X(NCRC,LL) = X(NCRC,LL) + CI*(FLAG1*XA(NORDP1,LL) - A(NORDP1)*(FLAG1-
1*FLAG2)*XA(NCRC2,LL)) + ICBS
IF (LL.EQ. NCRP3) GO TO 15
GC TO 4
15 CC 16 II = 1,ACRC
18 X(NCRC,LL) = X(NCRC,LL) + ET*(AI(I)*CI(I,L))
X(NCRC,LL) = X(NCRC,LL) + CI*AXCRP1 + TI*(FLAG1-FLAG2)*ICBS
4 CONTINUE
CC 12 II = 1,ACRP2
CC 12 KK = 1,ACRP2
IF (APS(X(II,KK)),.EQ. 1,F+1)) CC TO 16
13 XA(II,KK) = X(II,KK)
L = L + 1
FLAG2 = FLAG1
3 FLAG1 = FLAG1 + CR
IF (ICFT.LE.1) CC TO 6
IF (L.EQ. MT) FFH = 7.
K = MT + (I-1) + 1
PP1 = PP1 + (MTA(I) - C(1,K)) =*2
8 CONTINUE
RETURN
16 CC LK = C
RETURN
CNE

SUBROUTINE matrix(j)
C
SUBROUTINE MATRIX USES THE INFORMATION GENERATED IN SOLUTION
TO FORM A SET OF ALGEBRAIC EQUATIONS WHICH CORRESPOND TO A
LEAST SQUARES CRITERION

COMMON R(21), BETA(21), DT, TLCS, CI 6, 1C, 1A, A(6), X(14, 15),
IAPAT(14, 14), VVEC(14), CC(14), BF, I, XV(14, 15), IC
COMMON ACBS, IPRAT, A1, BORS, NORDP1, NORDP2, NCR, ACRP3, NCRP2, NCRP2, FTR

IF (J, NE, MIN + 1) CC TO 1
CC 2 I = 1, NCRP2
BVEC(1) = 0.
CC 2 K = 1, NCRP2
2 AMAT(1, K) = 0.
1 CC 3 J = 1, NCRP2
BVEC(1) = BVEC(1) + (BETA(J) - X(1, NORDP2)) * X(1, J)
CC 3 K = 1, NCRP2
3 AMAT(1, K) = AMAT(1, K) + X(1, J) = X(1, K)
RETURN
END

SUBROUTINE GAUSK(K)
SUBROUTINE GAUSK SOLVES THE EQUATIONS GENERATED IN MATRIX FOR
THE NEW SET OF PARAMETERS
PROCEDURE IS USED
COMMON R(21), BETA(21), OT, TLCS, CI 6, 1C, 1A, A(6), X(14, 15),
IAPAT(14, 14), VVEC(14), CC(14), BF, I, XV(14, 15), IC
COMMON ACBS, IPRAT, A1, BORS, NORDP1, NORDP2, NCR, ACRP3, NCRP2, NCRP2, FTR
DIMENSION AM(14, 14), X(14, CI), IC
EQUIVALENCE (AMAT(1, 1), AA(1, 1)), (B(1), BVEC(1)), (CC(1), C(1))
IF (IAPAT, NE, 1) CC TO 14
WRITE(6, 79)
79 FORMAT(5F5, AMAT)
CC 83 I = 1, NCRP2
83 WRITE(6, 83) (AA(I, J), J = 1, NCRP2)
80 FORMAT(1X, 7E17.6)
WRITE(6, 80) (B(I), J = 1, NORDP2)
81 FORMAT(5H30FEC/(1X, 7E17.6))
10  NA = N - 1
   CC 5  I = 1, NA
   J = N + 1 - 1
   L = J - 1
   CC 20  II = 1, L
   IF(ABS(AA(J,J)) .GE. ABS(AA(II, J))) GC TO 25
   CC 21  LL = 1, N
   AS = AA(J, LL)
   AA(J, LL) = AA(II, LL)
20  AA(II, LL) = AS
   AS = P(II)
   B(II) = B(J)
   B(J) = AS
20  CONTINUE
   CC 6  K = 1, L
   FACT = AA(K, J) / AA(J, J)
   CC 7  K = 1, J
   AA(K, N) = AA(K, P) - FACT * AA(J, K)
   6  P(K) = P(K) - FACT * B(J)
   5  CONTINUE
   CC 11  I = 2, N
   SUM = P(1)
   P = I - 1
   CC 9  J = 1, P
   9  SUM = SUM - AA(I, J) * G(J)
11  G(1) = SUM / AA(1,1)
   IF(IPRT .EQ. 1) WRITE(6, B2) (G(I), I = 1, N, P2)
82  FORMAT(31: E/E, 7E17.8))
RETURN
END

SUBROUTINE REGEN(L)
   SUBROUTINE REGEN UPDATES HLL OUTPUT AND ITS DERIVATIVES FOR THE
C     NEXT ITERATION
    CC CCM C R (21) , R ETA (21) , C 1 , TCPS , C ( 6 . 1 0 ; C ) , A ( 8 ) , X ( 1 4 , 1 5 ) ,
    1  AFAT (14, 14) , MV E (14) , CC (14) , F F I , XA (14, 15) , I C
    CC CRK R KOB S , IP CM 1 , A T 3 , M D R , M CR P 1 , M CR P 2 , N K , N C R P 3 , M CR P 2 , M C R P 2 , M T I
    X 4 J = 1 , M C R C
    4 G ( J , L ) = X ( J . M C R P 3 )
    CC 1 J = 1 , M C R C
    CC 1 1 = 1 , M C R P 2
    X C ( J , L ) = C ( J , L ) + C C ( I ) = X ( J , I )
    IF ( L . NE. M T F + M T F + I ) G 0 1 0 6
    IF ( I F F N T . E Q . 1 ) WRITE (7, 5)
    5 F O R M A T ( 4 P L , 5 X , 2 I C 1 , 8 X , 2 I C 2 , 8 X , 2 I C 3 , 8 X , 2 I C 4 , 8 X , 2 I C 5 , 8 X , 2 I C 6 , 8 X ,
    1 2 I C 7 , 8 X , 2 I C 8 , 8 X , 2 I C 9 , 8 X , 2 I C 1 0 )
    6 IF ( I F F N T . E Q . 1 ) WRITE (11, 2 1 1 , ( C ( I , L ) , 1 = 1 , M C R P )
    2 F O R M A T ( 1 5 , 1 1 , I F G , 4 )
    R E T U R N
    E N D

C     INPUT DATA
    2    1 1 0 0    . 1    . 5    1 5
    1  . 7 5    2  . 5

    1  . 1  . 1  . 1  . 1  . 1  . 1  . 2  . 5  . 2  . 5
QUASILINEARIZATION

CHANGE
C STATE VARIATIONS DEFINED IN TERMS OF THE CHANGE IN THE
C PARAMETERS
C THIS PROGRAM DOES NOT INCLUDE DEAD TIME
COMMON R(203),BETA(203),DT,TOES,CT,6,12,14,AT(7),X(13,14),
1AP(13,13),BVEC(13),CC(13),PHI,XA(13,14),
COMMON ACUS,IPRT,KT,NCRP,NCRP1,NCRP2
25 ITR = 1
DCM = 1.3
READ(5,7) NORD,NCES,CT,TOPS,IPRT
7 FORMAT(21L2,2E15.7)
C NORD = ORDER OF MODEL EQUATION
C NCES = NUMBER OF OBSERVATIONS
C CT = INTEGRATION DELTA T
C TOES = TIME BETWEEN OBSERVATIONS
C IPRT = PRINT CONTROL
C IPR = : ONLY THE ANSWER IS PRINTED AT EACH ITERATION
C IPR = 1 PRINT INFORMATION PRINTED AT EACH ITERATION
C ITR = ITERATION NUMBER
C PH = VALUE OF SUM OF THE SQUARES.
C IF(NORD.LE.3) CALL EXIT
ACRP1 = NORD + 1
ACRP2 = NORD - 1
ACR = 2 * NORD
ACRF1 = ACR + 1
ACRF2 = ACR + 2
CALL XII
CALL ASSUM
2 CALL SCRN(1)
CALL CAUSS(ACRP1)
C A PREASSIGNED PORTION OF THE CALCULATED STEP SIZE IS TAKEN
C THE PORTION OF THE STEP SIZE CM IS VARY BY VARYING DCM
DCM = DCM / 1.35
CH = 1. - DEC
CH 51 I = 1, NCRC1
51 CC(I) = CC(1) + CH
CALL SCILN(2)
CC 56 I = NCRC1, NCRC1
J = 1 - NCRC
50 CC(I) = CC(I) + A(J)
C CHECK FOR CONVERGENCE
CC 15 I = NCRC1, NCRC1
J = 1 - NCRC
15 IF(APS((A(J) - CC(I)) / (A(J) + CC(I))), GT, F, 1) GO TO 1
WRITE(6, 3)
3 FORMAT(9H,SOLUTION)
WRITE(A, 3) ITR, PHI
CC 16 I = 1, NCRC
16 WRITE(6, 4) I, A(I)
WRITE(6, 2) A(1, NCRC1)
CC TO 25
1 CC 16 I = 1, NCRC
J = NCRC + 1
13 A(I) = CC(J)
A(1, NCRC1) = CC(1, NCRC1)
WRITE(5, 3) ITR, PHI
3 FORMAT(11H,ITERATION 15/7 PHI = 815.5)
CC 22 I = 1, NCRC
22 WRITE(6, 4) I, A(I)
4 FORMAT(2H,CALL, 31 = 14.4)
WRITE(6, 2) A(1, NCRC1)
20 FORMAT(9H,SATN = 14.4)
IF(ITR . LE. 25) CC TO 25
ITR = ITR + 1
CC 1C 2
60 STOP
END
SUBROUTINE XIN
C
This subroutine generates data from the model equation.
C
COMMON (21), RELA(21), DT, TCRS, CT(6,1), A(7), X(13,14),
1 JACAT(13,13), POCC(13), CC(13), PH, XA(12,14)
C
COMMON XCRS, IPRT, AT, NODR, NCRP1, NODRP1, NCR, NCRP1, NCRP2
C
DIMENSION AA(21), CA(13), CS(14)
C DC 3 I = 1,10
3 AA(I) = 0.
C
NC = ORDER OF DATA
C
AA = PARAMETERS USED TO GENERATE DATA
C
A(I) THRU A(6) = DESIRED PARAMETERS RELATED TO TIME CONSTANTS
C
OF THE SYSTEM
A(6) = GAIN OF THE SYSTEM
A(1) = DEAD TIME OF SYSTEM IN SAMPLING PERIODS
C
This program requires a random number generator
C
ANI = FACTOR TO ADJUST HEIGHT OF UNEVALUATED INLET NOISE
C
ANC = FACTOR TO ADJUST HEIGHT OF UNEVALUATED OUTPUT NOISE
C
ANC = FACTOR TO ADJUST HEIGHT OF MEASURED NOISE ON OUTLET
C
NS = NUMBER OF SIGNIFICANT FIGURES IN DATA

READ(5,1) NC, (AA(I), I = 1,10)
10 FORMAT(116,1CF5.2)
READ(5,9) ANI, ANC, ANF, NS
9 FORMAT(3F10.5,112)
WRITE(6,3) ANI, ANC, ANF
3 FORMAT(12H INPUT NOISE, F10.5, EX, 12H OUTPUT NOISE, F10.5)/
12H MEASURED INPUT NOISE, F10.5)

XC = .64
YC = . 6 2
ZP = RAND(XC, YC)
XC = YC
NCRP1 = XC - 1
NT = 1000/DT .5
NCRSP1 = NCRS + 1
C  11  J = 1, ACSF1
    FORCING  FUNCTION  IS A TVALED SIN.
    R(I) = (SIN(FLOAT(I) * 38) / 10.) * EXP(-FLOAT(I) * .1)
    JJ = 17. * NS + F(I)
 11  R(I) = FLOAT(JJ) / 1. * NS
    CC 91  I = 1, ACSF1
 60  R(I) = R(I) + RAND(XC, YC) * ANI #2. - ANY + RAND(CX, YC) * ANE #2. - ANY
    CC 88  I = 1, NC
    CE(I) = 5.
 8  CA(I) = 5.
    CC 1  I = 1, ACSF1
    FLAG = R(I)
    CR = (R(I+1) - R(I)) / FLOAT(I)
    BETA(I) = CA(I) + RAND(XC, YC) * ANE #2. - ANY
    JJ = 17. * NS + BETA(I)
    BETA(I) = FLOAT(JJ) / 1. * NS
    CC 1  I = 1, NT
    IF(AE, EC, 1) GE 1052
    CC 5  I = 1, ACSF1
    KR = 11 + 1
 5  CA(I) = CA(I) + CT + CR(KK)
 93  CONTINUE
    CC 4  I = 1, AC
 4  CA(AC) = CA(AC) + CT * (-AA(11) * CA(I1))
    CA(AC) = CA(AC) + (AA(9) + FLAG) * CT
    CC 5  I = 1, AC
 50  CH(I) = CA(I)
 1  FLAG = FLAG + CR
    CC 12  I = 1, ACSF1
    R(I) = (SIN(FLOAT(I) * 38) / 10.) * EXP(-FLOAT(I) * .1) +
    RAND(XC, YC) * ANI #2. - ANY
    JJ = 17. * NS + S(I)
 13  R(I) = FLOAT(JJ) / 1. * NS
    WRITE(6, 0) NC
SUBROUTINE ASSUME
SUBROUTINE ASSUME GENERATES INITIAL SOLUTION
C
C       COMMON R(2N),BETA(2N),P(I),NCRS,CI(6,12),A(7),X(13,14),
1       IAMAN(13,13),BBEC(13),CC(13),PHI,XA(13,14)
C
C       COMMON NCR,B,RM1,N1,NCRP1,NCRPM1,NCR,NCRBP1,NCRBP2
C
C
C
C       C(1,1) = 0.
C       CR = 0.
C       NCRSF1 = NCRS + 1
C       CI 3 I = 1,NCRSF1
C       FLAG = 2(I)
C       CR = (R(I+1) - R(I))/FLOAT(NT)
C       CI 1 LL = 1,NT
53 \text{CB} = C(\text{NCNP}, L)
\text{CC} 4 11 = 1, \text{NCRC}
4 \text{CB} = \text{CB} + \text{DT} * (-A(11) = C(11, L))
\text{C}(\text{NCRC}, L+1) = \text{CB} + A(\text{NCNP}) + \text{FLAG} * \text{DT}
\text{IF} (\text{NCRC} \neq 7) \text{CC} 1C 7
\text{CC} 8 11 = 1, \text{NCRCF1}
\text{KK} = 11 + 1
8 \text{C}(11, L+1) = \text{C}(11, L) + \text{DT} * \text{C}(\text{KK}, L)
7 \text{IF} (L \neq 1) \text{CC} 6
\text{IF} (\text{IFNT} \neq 1) \text{WRITE}(6, 5)
5 \text{FORMAT}(415) \text{L}, 5X, 2HC1, 9X, 2HC2, 8X, 2HC3, 8X, 2HC4, 8X, 2HC5, 8X, 2HC6, 8X, 12HC7, 8X, 2HC8, 8X, 2HC9, 8X, 3HC1
6 \text{IF} (\text{IFNT} \neq 1) \text{WRITE}(6, 9) \text{L}, (C(F, L) = 1, \text{NORD})
9 \text{FORMAT}(15, 11F12.4)
\text{FLAG} = \text{FLAG} + \text{CB}
1 L = L + 1
\text{LL} = \text{AT} * (1 - 1) + 1
\text{PHI} = \text{PHI} + (\text{BETA}((1) - C(11, L))) * \text{LL}
3 \text{CONTINUE}
\text{WRITE}(6, 10) \text{PHI}
\text{RETURN}
\text{END}

\text{SUBROUTINE} \text{SCAL}(10F1)
\text{COMMON} R(20,:), \text{BETA}(2,:), \text{NT}, \text{CCS}, C(6, 12, C), A(7), X(13, 14),
\text{C}
\text{SUBROUTINE SOLUTION GENERATES THE FUNDAMENTAL SOLUTIONS}
\text{C}
\text{ELSE IF THE FORM OF THE STATE VARIABLES NO PARTICULAR SOLUTION}
\text{C}
\text{IS NECESSARY}
1 \text{IAAF}(13, 13), \text{PVAC}(13), \text{CC}(13), \text{PHI}, XA(13, 14)
\text{COMMON} NCRS, IPRT, AT, NO D, NORDP1, NORDP2, NCR, NCNP1, NCNP2
L = 1
\text{CC 2 I = 1, NCREF1}
\text{CC 1 J = 1, NCREF2}
\text{XA(I, J) = 0.}
1 \( X(1,J) = 0 \).
1  \( XA(1,1) = 0 \).

2 \( X(1,1) = 1 \).
6 CC E J = 1,ACPS
4 IF(ICFT .NE. 1) CC TO 1;
4 IF(IFMAT .NE. 1) CC TO 11
4 WRITE(5,66) J
66 FORMAT(12C,COBSERVATION 15)
6 CC 5C F = 1,ACRP1
65 WRITE(6,61) (X(A,K),A = 1,ACRP2)
61 FCRMAT(1X,7F17.5)
11 CALL FAIRX(J)
10 GR = (R(J+1) - R(J))/FLOCAT(N1)
9 RR = R(J)
6 CC 3 K = 1,NT
8 IF(ICFT .EQ. 2) CALL RFSEQ(L)
6 CC 4 I = 1,ACRP1
6 CC 5 AM = 1,ACRP1
5 X(RH,1) = X(RH,1) + CT * XA(4N+1,1)
6 CC 6 KK = 1,ACRP2
6 II = ACRP + RR
6 X(NCRC,1) = X(NCRC,1) + DT * ((-A(KK)*XA(KK,1)) + (-C(KK,L)*XA(J1,1)))
6 x(NCRC,1) = X(NCRC,1) + DT * (RR + XA(NCRCP1,1))
4 CONTINUE
6 CC 13 II = 1,ACRP1
6 CC 12 KK = 1,ACRP1
13 XA(11,KK) = X(II,KK)
4  L = L + 1
3 RR = RR + DR
4 IF(ICFT .EQ. 1) CC TO 8
4 IF(J .EQ. 1) PHI = 0.
5 K = NT * (J-1) + 1
4 PHI = PHI + (BETA(J) - C(J,K))*2
8 CONTINUE
SUBROUTINE MATRX(J)
SUBROUTINE MATRX USES THE INFORMATION GENERATED IN SOLUTICA
TO FORM A SET OF ALGEBRAIC EQUATIONS WHICH CORRESPOND TO A
LEAST SQUARES CRITERION.
COMM P(26), BETA(26), IX, TCBS, CI, 6, R(7), A(7), X(13,14),
1. AMAT(13,13), BVEC(13), CC(13), PI, XA(13,14),
COMM NCRS, IPRNT, KI, NORD, NCRP1, NORDP1, NCR, NCRP1, NCRP2
L = M1 = (J - 1) + 1
IF ( J .NE. 1 ) CC TO 1
CC 2 I = 1, NCRP1
BVEC(1) = 1.
CC 2 K = 1, NCRP1
2. AMAT(1,K) = J.
1. CC 3 I = 1, NCRP1
BVEC(1) = BVEC(1) + (BETA(J) - C(1,L) )
CC 3 K = 1, NCRP1
3. AMAT(1,K) = AMAT(1,K) + X(1,1) * X(1,K)
RETURN
END

SUBROUTINE GAUSS(N)
SUBROUTINE GAUSS SOLVES THE EQUATIONS GENERATED IN MATRX FOR
THE NEW SET OF PARAMETERS
RCN INTERCHANGE IS USED.
COMM P(26), BETA(26), PI, TCBS, CI, 6, R(7), A(7), X(13,14),
1. AMAT(13,13), BVEC(13), CC(13), PI, XA(13,14)
COMM NCRS, IPRNT, KI, NORD, NCRP1, NORDP1, NCR, NCRP1, NCRP2
DIMENSION A(12,13), R(13), C(13)
EQUIVALENCE (AMAT(1,1), AA(1,1)), (R(1), BVEC(1)), (CC(1), C(1))
IPRNT = 1
IF(IPRNT .NE. 1) GO TO 10.
WRITE(6,8)(AA(I,J),J = 1,N(NP1)), I = 1,N(MP1))  
80 FORMAT(5,LA,AT/(1X,CT17.8))  
WRITE(6,81) (E(I), I = 1,N(MP1))  
81 FORMAT(5,6V,CE/(1X,7F17.8))  
15 NN = N - 1  
CC 5 I = 1,NN  
J = N+1 - 1  
L = J - 1  
CC 20 II = 1,L  
IF(AES(AA(J,J)), .GF. AES(AA(II,J))) GO TO 20  
CC 21 LL = 1,K  
AS = AA(J,LL)  
AA(J,LL) = AA(II,LL)  
21 AA(II,LL) = AS  
AS = E(I1)  
B(I1) = E(J)  
E(J) = AS  
20 CONTINUE  
CC 6 K = 1,L  
FACT = AA(K,J) /AA(J,J)  
CC 7 P = 1,J  
AA(K,P) = AA(K,F) - FACT *AA(J,P)  
6 B(K) = B(K) - FACT * B(J)  
5 CONTINUE  
G(I) = B(I) /AA(I,1)  
CC 11 I = 2,N  
SLV = B(I)  
F = I - 1  
CC 9 J = 1,F  
9 SLV = SLV - AA(I,J) * G(J)  
11 G(I) = SLV /AA(I,1)  
IF(FFAT .NE. 1) GO TO 12  
WRITE(6,62) (G(I), I = 1,NCRP1))  
62 FORMAT(3F6CC/(1X,7F17.8))
SUPRO UNE REGUM(L)

SUBROUTINE RECFA (UPDATES THE OUTPUT AND ITS DERIVATIVES FOR THE NEXT ITERATION)

C

COMPRA A(2, 1), BETA(2, 1), DT, HT, HS, C(6, J, L), A(7), X(13, 14),
1AMAT(13, 13), GVEC(13), CC(13), H1, XA(13, 14)

COMPRA ACRS, IPRTAT, A2, NORDA, NORDA1, NORDA2, NORDA1, NORDA2

CC 1 J = 1, NCR

CC 1 I = 1, NCR

1 C(J, L) = C(J, L) + CC(I) = X(J, 1)

IF(L .NE. 1) CC TC 6

IF(IFINT .EQ. 1) WRITE(4, 5)

5 FORMAT(416, 5X, 2IC), AX, 2HC2, EX, 2HC4, EX, 2HC6, EX, 2HC8, EX, 2HC10)

12HC7, EX, 2HC9, EX, 2HC11)

6 IF(IFINT .EQ. 1) WRITE(6, 2) L, (C(1, L), 1=1, NCRD)

2 FORMAT(15, 11F10.4)

RETURN

END

C

INPUT DATA

2 50 .01 .25 2.5
2 2.5 2.5

100. 100. 100. 100. 100. 100. 100. 100.
QUASILINEARIZATION

MARQUARDT
QUASILINEARIZATION

INCORPORATION OF MARQUARDT'S METHOD TO INCREASE PARAMETER SPACE

CONVERGENCE

THIS PROGRAM DOES NOT INCLUDE DEAD TIME

COMMON R(2,1), BETA(2,1), DT, TCR(6,1), A(7), X(13,14),
1AMAT(13,13), RVL(13), CC(13), PH1, XA(13,14), ITR
COMMON NCRS, IPRT, AT, NCRO, NCRP1, NCRP2, NCRP1, NCRP2, NCRP2
DIMENSION SCC(12)

25 ITR = 1

NCR = ORDER OF MODEL EQUATION

NCBS = NUMBER OF OBSERVATIONS

CT = INTEGRATION CELTA T

TDE = TIME BETWEEN OBSERVATIONS

IFRT = PRINT CONTROL

IFRT = 0 CALY THE ANSWER IS PRINTED AT EACH ITERATION

IFRT = 1 MORE INFORMATION PRINTED AT EACH ITERATION

ITR = ITERATION NUMBER

PHI = VALUE OF SUM OF THE SQUARES.

READ(5,7) NCR, NCBS, CT, TDE, IPRT

7 FORMAT(21L,2F15.6,10)

IF(NCR.LT.0) CALL EXIT

NCRP1 = NCR + 1

NCRP1 = NCR - 1

KCR = 2 * NCR

NCRP1 = NCR + 1

NCRP2 = NCR + 2

CALL XIA

CALL ASSURE

CC 50 I = 1, NCR

CC(1) = 0.

50 SCC(I) = 0.

CC 51 I = NCRP1, NCRP1

J = 1 - NCR
CC(I) = A(I)
51 SCC(I) = A(I)
2 CALL SCLN(1)
CALL GAUSS(ACRP1)
CC 52 I = 1, NCRF1
SCC(I) = SCC(I) + CC(I)
52 CC(I) = SCC(I)
CALL SCLN(2)

C CHECK FOR CONVERGENCE
DC IF I = NCREP1, ACRP1
J = I - NCRF
15 IF(ABS((A(J) - CC(I)) / (A(J) + SCC(I))).GT.8.E-61) CC TC 1
WRITE(*, 36)
36 FORMAT(102,10E15.5)
WRITE(*, 3) ITR, Phi
CC 16 I = 1, NCRF
16 WRITE(6, 4) I, A(I)
WRITE(6, 26) A(ACRP1)
CC TC 25
1 CC 16 I = 1, NCRF
J = NCRF + 1
16 A(I) = CC(J)
A(ACRP1) = CC(ACRP1)
WRITE(6, 3) ITR, Phi
3 FORMAT(11F10.6, 7E10.5)
WRITE(6, 3) ITR, Phi
CC 22 I = 1, NCRF
22 WRITE(6, 4), A(I)
4 FORMAT(29F10.6, 3E10.5)
WRITE(6, 29), A(ACRP1)
29 FORMAT(11F10.6, 7E10.5)
IF(ITR .EQ. 53) CC TC 25
ITR = ITR + 1
CC TC 2
6C STOP
END

SUBROUTINE XIA

THIS SUBROUTINE GENERATES DATA FROM THE MODEL EQUATION.

COMMON B(20), beta(20), DT, tc, cs, A(7), X(13,14),
IAMP(13,13), PVEC(13), GC(13), PHI, XA(13,14), IC
COMMON ABCS, IPTAI, AT, AORD, WORDP, WORDP1, HCR, WORDP1, HCRP2

DIMENSION AAM(I), CA(15), CR(I)

CC 3 I = 1, 12

3 AAM(I) = J.

C NC = ORDER OF DATA
C NC = ORDER OF DATA
C AA = PARAMETERS USED TO GENERATE DATA
C A(I) THRU A(N) = DESIRED PARAMETERS RELATED TO TIME CONSTANTS
C OF THE SYSTEM
C A(5) = GAIN OF THE SYSTEM
C A(10) = TIME OF SYSTEM IN SAMPLING PERIODS

READ(5, 1) NC, (AAM(I), I = 1, 12)

10 FORMAT(12(5X, F5.2))

C THIS PROGRAM REQUIRES A RANDOM NUMBER GENERATOR
C AAM = FACTOR TO ADJUST HEIGHT OF UNMEASURED INLET NOISE
C AAM = FACTOR TO ADJUST HEIGHT OF UNMEASURED OUTLET NOISE
C AAM = FACTOR TO ADJUST HEIGHT OF MEASURED NOISE ON OUTLET
C NS = NUMBER OF SIGNIFICANT FIGURES IN DATA

READ(S, 9) AAM, AAM, AAM, AAM

9 FORMAT(3(15, 5X, I10))

WRITE(6, 3) AAM, AAM, AAM

3C FORMAT(12H INPUT ACISE, F1.5, 5X, 12H OUTPUT NOISE, F1.5)
121H. MEASURED INLET NOISE, F1.5)

XC = .84
YC = .63

ZP = READ(XC, YC)

XC = YC

AAM1 = NC - 1
\[ F \cdot T = T \cdot C \cdot R \cdot S / i \cdot H + 0.5 \]
\[ K \cdot C \cdot B \cdot S \cdot F = G \cdot ! \cdot S + 1 \]
\[ C \cdot C \cdot 1 \cdot I = 1 \cdot N \cdot C \cdot P \cdot S \cdot 1 \]
\[ F \cdot L \cdot A \cdot G \cdot I = X \cdot C \cdot I \cdot X \cdot C \cdot I \cdot N \cdot I \cdot S \cdot A \cdot [1 \cdot A \cdot F \cdot I \cdot S \cdot I \cdot M \cdot I \cdot T \cdot C \cdot S \cdot 1] \]
\[ R \cdot (I) = (S \cdot I \cdot N \cdot (F \cdot L \cdot C \cdot A \cdot T \cdot (I)) * 3.8) * 1.2 * \exp(-F \cdot L \cdot A \cdot O \cdot T \cdot (I)) * .1 \]
\[ J \cdot J = 1.0 * N \cdot S \cdot * R \cdot (I) \]
\[ 11 \ R \cdot (I) = F \cdot L \cdot C \cdot A \cdot T \cdot (J \cdot J) / 1.0 * N \cdot S \]
\[ C \cdot C \cdot 8 \cdot C \cdot I = 1 \cdot N \cdot C \cdot P \cdot S \cdot 1 \]
\[ 8 \cdot 0 \cdot R \cdot (I) = R \cdot (I) + R \cdot A \cdot N \cdot D \cdot (X \cdot C \cdot Y \cdot C) * A \cdot N \cdot I \cdot N \cdot Z \cdot 2. - A \cdot N \cdot I \cdot N \cdot D \cdot (X \cdot C \cdot Y \cdot C) * A \cdot N \cdot M \cdot Z \cdot 2. - A \cdot N \cdot W \]
\[ C \cdot C \cdot 8 \cdot I = 1.0 * N \cdot C \cdot P \cdot S \cdot 1 \]
\[ 8 \cdot C \cdot A \cdot (I) = . \]
\[ C \cdot C \cdot I = 1 \cdot N \cdot C \cdot P \cdot S \cdot 1 \]
\[ F \cdot L \cdot A \cdot G \cdot I = . \]
\[ 8 \cdot C \cdot A \cdot (I) = . \]
\[ C \cdot C \cdot I = 1.0 * N \cdot C \cdot P \cdot S \cdot 1 \]
\[ F \cdot L \cdot A \cdot G \cdot I = . \]
13 F(I) = FLOAT(JJ) / 13.4415
WRITE(6,61) NC
61 FORMAT(1H1, 39X, 41ECATA/14HRORDER OF DATA IS )
CC 61 I = 1,8
61 WRITE(6,62),AA(I)
62 FORMAT(2F4.1,2F = F7.4)
WRITE(6,63) AA(8)
63 FORMAT(2F4.1,2F = F7.4)
WRITE(6,64) AA(10)
64 FORMAT(13F4) REACT Time = F7.4)
WRITE(6,2)(R(I), BETA(I), I = 1, NOBS)

2 FORMAT(1X, 2F25.6)
RETURN
END

SUBROUTINE ASSUME

SUBROUTINE ASSUME GENERATES INITIAL SOLUTION
COMMON R(I), BETA(I), DT, TOL, C(I,12), A(7),X(13,14),
1AMAT(13,13), BVEC(12), CC(13), PH1, XA(13,14), XIC
COMMON NCBS, IP, NT, NCNT, NCRP1, NCP1, NC, NC2, NC2P, NCRP2

12 FORMAT(7F13.5)
13 FORMAT(23H0PHI FOR ASSUMED SOLUTION FILE.5)
L = 1.
PFI = .

C A = ASSUMED PARAMETERS
C THE PARAMETERS ARE READ IN CONSECUTIVE FIELDS WITH THE LAST
C BEING THE GAIN

REAL(A(I),I = 1, NCRP1)
CC 2 I = 1, NCRE
2 CC I = 1, NCRES
FLAG = R(I)
CR = (R(I+1) - R(I))/F(VAT(N1))
CC 1 LL = 1, NT
50 CR = C(NCRD,L)
CC 4 II = 1, NCRD
4 CR = CE + DT = (A(I1) * C(I1,L))
C(NCRD,L+1) = CR + A(NCRDP1) * FLAG * DT
IF(A(NCRD,EQ. 1) CC TO 7
CC 8 II = 1, NCRDMP
KK = II + 1
8 C(I1,L+1) = C(I1,L) + E1 * C(KK,L)
7 IF( L .NE. 1) CC IL 6
   IF(IPRM .EQ. 1) WRITE(6,5)
5 FORMAT(4H1, L, 9, 2HC1, 8X, 2HC2, 8X, 2HC3, 8X, 2HC4, 8X, 2HC5, 8X, 2HCC, PX,
   9 12HC7, 8X, 2FC8, 8X, 2FC9, 8X, 3HC1, )
6 IF(IPRM .EQ. 1) WRITE(6,9), L, (C(M,L), M=1, NORD)
9 FORMAT(15,11FLC,4)
   FLAG = FLAG + CR
1 L = L + 1
   LL = M1 + (I-1) + 1
   PH1 = PH1 + (PHETA(I) - C(1, LL)) * 2
3 CONTINUE
   WRITE(6,1) PH1
   RETURN
END

SUBROUTINE SOLN(ICPI)
SUBROUTINE SOL\(N\) GENERATES THE FUNDAMENTAL SOLUTIONS
CCMPCA R(2,1), ETA(23), DT, TOPS, C( 6, L=1), AT( 7), X(13,14),
1APAT(13,13), BYEC(13), CC(13), PH1, XA(13,14), IC
CCMPCA NCRS, IPRA, AT, NORD, NCRDP1, NORDP1, NCR, NCRDP1, NCRP2
L = 1
CC 2 I = 1, NCRDP1
CC 1 J = 1, NCRP2
XA(I,J) = 3.
1 \( x(i,j) = 0 \)
\( x_a(i,1) = 0 \)
2 \( x(i,1) = 1 \)
CC 8 J = 1,ACRF
IF(ICFT.NE. 1) CC TO 11
IF(IFANT .NE. 1) CC TO 11
WRITE(6,'(A,')) J
60 FORMAT(1X,'CCRTAT(12):COMPARISON 15)
CC 9: M = 1,ACRF1
50 WRITE(6,61) (X(K,1),K = 1,NCRP2)
61 FORMAT(1X,3F17.5)
11 CALL MATRX(J)
12 OR = (R(J+1) - R(J))/PLOAT(X1)
|  RR = R(J)
|  CC 3 K = 1,AT
|  IF(ICFT.EQ.2) CALL REGEM(I)
CC 4 I = 1,NCRP2
CC 5 NN = 1,ACRF1
5 X(KK,1) = X(KK,1) + CI * XA(KK+1,1)
CC 6 KK = 1,ACRF
11 = NCRD + KK
6 X(NCRD,1) = X(NCRD,1) + DI * ((-A(KK)*XA(KK,1)) + (-C(KK,L)*XA(I1,I1)))
| X(NCRD,1) = X(NCRD,1) + DI * (RR * XA(ACRP1,1))
| IF(I1 .EQ. NCRP2) CC TO 15
CC 10 4
15 CC 1E II = 1,ACSC
18 X(NCRD,1) = X(NCRD,1) + DI * (A(I1)*C(I1,L))
4 CONTINUE
CC 12 II = 1,ACRF1
CC 12 KK = 1,ACRP2
13 XA(I1,1) = X(I1,KK)
L = L + 1
3 RR = RR + OR
IF(ICFT.EQ.1) CC TC 8
IF(J .EQ. 1) PH1 = C
K = NT + (J-1) + 1
PH1 = PH1 + (BETA(J) - C(I,K)) * 2
8 CONTINUE
RETURN
END

SUBROUTINE MATRIX(J)

SUBROUTINE MATRIX USES THE INFORMATION GENERATED IN SOLVE
TO FORM A SET OF ALGEBRAIC EQUATIONS WHICH CORRESPOND TO A
LEAST SQUARES CRITERION
THE AMAT AND EVEC CORRESPOND TO THOSE USED IN GARNAUDTS
METHOD
CMMCM R(2), BETA(2), DT, ICES, C(6,1..6), A(7), X(13,14),
IAMA(13,13), BVEC(13), CC(13), PH1, XA(13,14), IC
CMMCM NCRS, IPRAT, AT, NCRD, KCRCP1, NCRDA1, NCR, NCRP1, NCRP2
DIMENSION B(13)
IF( J .NE. 1) CC TO 1
C
V = INITIAL VALUE OF LAMBDA
IF( IC, EC, 1 ) REZG(5,2) V
20 FCRMAT(12,5)
CC 2 I = 1, NCRP1
BVEC(I) = C.
CC 2 K = 1, NCRP1
2 AMA(1,I,K) = C.
1 CC 3 I = 1, NCRP1
BVEC(I) = BVEC(I) + (BETA(J) - X(1,NCRP2)) * X(1,1)
CC 3 K = 1, NCRP1
3 AMA(1,K) = AMA(1,K) + X(1,1) * X(1,K)
IF( J .NE. NCRS) RETURN
CC 11 I = 1, NCRP1
R(I) = ..
CC 1C L = 1, NCRP1
10 P(I) = -AMAT(I,I) * CC(L) + B(I)
11 BVEC(I) = BVEC(I) + B(I)
CC12 I = 1, NCRP1
12 AMAT(I,I) = AMAT(I,I) + V

CHANGE VALUE OF LAMBDA FOR NEXT ITERATION
V = V/1.
RETURN
END

SUBROUTINE GAUSS(N)

SUBROUTINE GAUSS SOLVES THE EQUATIONS GENERATED IN MATRIX FOR
THE NEW SET OF PARAMETERS

REAL K(R(2),), BETA(R(2,)), DT, TCBS, C(6,1C,2), A(7), X(13,14),
1AMAT(13,13), BVEC(13), CC(13), PHI, XA(13,14), IC
COMMON NCBS, IPRMT, KT, NCRP, NCRP1, NCRP2, NCR, NCRP1, NCRP2
DIMENSION AA(12,12), B(13), C(13)
EQUIVALENCE (AMAT(1,1), AA(1,1)), (B(I), BVEC(I)), (CC(I), C(I))
IPRMT = 1
IF(IPRMT .NE. 1) GC TO 1?
WRITE(6,93)((AA(I,J), J = 1,NCRP1), I = 1,NORP1)
80 FORMAT(5E14.8)
WRITE(6,81) (B(I), I = 1,NORP1)
81 FORMAT(5E14.8)

10 AN = N - 1
CC 5 I = 1, AN
J = N + 1
L = J - 1
CC 21 II = 1, L
IF(ABS(AA(J,J)) .GE. ABS(AA(II,II))) GC TO 23
CC 21 LL = 1, A
AS = AA(J,LL)
AA(J,LL) = AA(II,LL)
21 AA(II,LL) = AS
AS = F(I)
B(I) = B(J)
B(J) = AS
20 CONTINUE
CC 6 K = 1,L
FACT = AA(K,J) / AA(J,J)
CC 7 N = 1,J
6 AA(K,N) = AA(K,N) - FACT * AA(J,N)
5 CONTINUE
CC 11 I = 2,N
SLW = B(I)
M = 1 - 1
CC 9 J = 1,N
9 SLW = SLW - AA(I,J) * G(J)
11 G(I) = SLW / AA(I,I)
IF(IPRNT . NE. 1) CC TO 12
WRITE(6,82)(G(I), I = 1,NCRP1)
62 FCPMAT(3F:CC/(1X, 7E17.8))
IPRNT = 0
12 RETURN
END

SUBCUTINE REGEN(L)
SUBCUTINE REGEN UPDATES THE OUTPUT AND ITS DERIVATIVES FOR THE
NEXT ITERATION
CCCPCH R(2), BETA(23), DT, TCHS, C( 6, 10*C), A( 7), X(13, 14),
1AMAT(13, 12), BVEC(12), CC(13), PH1, XA(13, 14), 1C
CCCPCH AORIS, IPRRN, AT, NORD, NCRP1, NCRP1, NCRP2, NCRP2
CC 4 J = 1,NCR
4 C(J,L) = X(J,NCRP2)
CC 1 J = 1,NCR
CC 1 I = 1,NCRP1
1 C(J,L) = C(J,L) + C(J) * X(J,L)
   IF(L .GE. 1) GC TC 6
   IF(1.LT.L .GE. 1) WRITE(6,5)
5 FORMAT(4HC, L, 5X, 2HC1, 8X, 2HC2, 8X, 2HC3, 8X, 2HC4, 8X, 2HC5, 8X, 2HC6, 8X, 2HC7, 8X, 2HC8, 8X, 3HC1)
6 IF(1.LT.L .GE. 1) WRITE(E,2) L, (C(I,L), I=1,NORD)
2 FORMAT(I9,11F12.4)
RETURN
END

C   INPUT DATA
   4    50    .01    .85
   4    2.5   2.5   2.5   2.5
   2.5   2.5   2.5   2.5
   .001   2.0   2.0   2.0   2.0
QUASILINEARIZATION

SOLUTION ASSUME
C  CUASILNEARIZATION
C  SOLUTION - ASSUME USES MODEL EQUATION TO UPDATE THE OUTPUT
C  AND ITS DERIVATIVES AT EACH ITERATION
C  THIS PROGRAM DOES NOT INCLUDE DEAD TIME
C
C CPMCA R(21), R(23), R(2), R(25), CC(6,13), A(7), X(13,14),
1 LAM1(13,13), DVEC(12), CC(13), PHI, XA(13,14)
C CPMCA NCRC, IPRT, AT, NCPO, NCRC1, NCRC1, NCRC1, NCRC1, NCRC1, NCRC1
C DIMENSION CSTR(12)
C
25 ITR = 1
C NCRC = ORDER OF MODEL EQUATION
C NCES = NUMBER OF OBSERVATIONS
C CT = INTEGRATION DELTA T
C TCES = TIME BETWEEN OBSERVATIONS
C IPNT = PRINT CONTROL
C IPNT = 3 ONLY THE ANSWER IS PRINTED AT EACH ITERATION
C IPNT = 1 MORE INFORMATION PRINTED AT EACH ITERATION
C ITR = ITERATION NUMBER
C PHI = VALUE OF SLP OF THE SLOPES.
C
97 IF (NCES.EQ. 0) CALL XIN
97 FORMAT(2112,2F10.2,I13)
97 IF (NCRC.EQ. 0) CALL EXIT
97 NCRC1 = NCRC + 1
97 NCRC1 = NCRC - 1
97 NCR = 2 * NCRC
97 NCRC1 = NCR + 1
97 NCRC1 = NCR + 2
97 CALL XIN
97 CALL ASSUME(I)
97 CC 52 I = 1, NCRC1
52 CC(I) = A(I)
97 CALL SCLN(I)
97 CC 52 I = 1, NCRC1
97 CSTR(I) = CC(I)
C SOLUTION RESTRICTED SO THAT CC1 AND CC2 ARE ZERO
DO 51 J = 1,NORCP1
51 AMAT(I,J) = AMAT(I+2,J+2)
50 BVEC(I) = BVEC(I+2)
PHIO = PHI
CALL GAUSS(NORDP1)
C CHECK FOR CONVERGENCE
DO 15 I = 1,NORC
15 IF(ABS((A(I)-CC(I))/(A(I)+.GGGG1))/.GT.0.001) GO TO 1
57 WRITE(6,33)
33 FORMAT(9HSOLUTION)
WRITE(6,3) ITR,PHI
DO 16 I = 1,NORC
16 WRITE(6,4)I,A(I)
WRITE(6,20)A(NORDP1)
GO TO 25
1 KCOUNT = 0
C IF THE VALUE OF PHI DOES NOT DECREASE THE STEP SIZE IS CUT IN
C HALF UNTIL IT DOES INCREASE OR OR 10 REDUCTIONS ARE MADE
11 DO 10 I = 1,NORCP1
10 A(I) = CC(I)
CALL ASSUME(2)
IF(PHI .LE. PHIC) GO TO 53
DO 54 I = 1,NORCP1
54 CC(I) = .5*(A(I)-CSTR(I))+CSTR(I)
KCOUNT = KCOUNT + 1
IF(KCOUNT .NE. 10) GO TO 11
DO 56 I = 1,NORCP1
56 A(I) = CSTR(I)
PHI = PHIO
GO TO 57
53 WRITE(6,3) ITR,PHI
3 FORMAT(110I4,15H PHI = E15.5)
DO 22 I = 1,NORC
22 WRITE(6,4)I,A(I)
WRITE(6,20)A(NORDP1)
7
7
1

W K ‘V £ 3 \ V ‘ 1'iV
S '

-isiok
a s m

M3 j s i o o

K

I

S

i

a.nnsviw

n

-

O

D

l

-j;j

d

L

N

V

U

I

O

l

V

b

H

K

O

O

W

3

I

1

M

V

(
U

O

I

N

3

m i

oi

I

< i

j

V

I

‘

S

I

3

« l

=

I

V

r.

C

'

‘

*

' J

\ *

\>

*

S

J

O

y

(! t ‘ 9 ) i L I tl M

‘J

*
I

-

‘ - T - j e

•'! V
i

d

(
W

O

£

M

I

‘

(

3

I

H

1

7

< i 0

)

l

4

s

V

W

'

d

a

)

=

ji b’O i o v H

' 1 9

O

V

J
1

S \ >

=

m v

= omv
= 1mv

L i d

d

S

I

-

I

I

(

2* m v i m i i ) i s/wan

)

V '

7

)

*

J

M

6

w

(

v

l

4

5

)

0

V

3

ox

H

3
V3 0
=
( 0
1
) V
JO \!IVJ = (S)V
H H I S A S D H 1 JO
c m v lib'S d o m v b v d
uddisn - m v
obHi
(n v
Vi Vl]
J 1 V d J N J 0 [ l i GO SO SHd i OAVb Vt l = VV
b

H

d

O

K

I

1

( i

'J V

S

‘

M

I

/

I

M

S

A

S

H

O

3

A

1

1

H-ll

V

£

I

i s n r a v 01 y o r o v j
i s n r a y 31 h j i o v h

■':!1 S A S

siwisnxjo

v

d

1

n o o o o n

S

I

-'J

i-ni 3,1 i s n r a y

i n 11 j x n y n s v o w M . n -io n o r i - i
m m
m m s v m n ho i o i m
d

H

‘

9

o o o o o

V . l V i l

.inmo

M

(

1 . 7

3

J

O

H

j

G

d

O

=

0

M

•o = ( i ) vv e
; 1 4I = I £ 0 3
( 11 ) ll 3 4 ( ■. I ) V J 1 ( 2 ) V V N J I S M0 A I a
ZddOK 4 I dVON 4 b-CJN 4T A 0 >. U \ 4 I dObUM 401! OK 4 1 M 4 L Mb dl 4SUOM M3HA33
(
[ 4 £ l ) V X 4 I Id 4 ( U ) 3 3 4 ( £ 1 ) 3-3 AH 4 ( £ I 4 CT ) I V A VI
4 ( v i 4 e x ) :<4 1 l ) v m v : t ‘ 9

•\S3Ilvn3H

3 3 00 A M I

) D 4 s d o i 4i : j M

A Odd

: : n y n u ‘ (

"Joaaoo

: z m

V i VO S3 l V b Fliv-O D J V I l T D d h f l S

mix

3

SI - 11

-

\

i

i

3

3

3

y

a

n

s

Q

K

3

dOiS

09

2 31 0 3
I + dl.1 = dil
S

Z

O
(

<

/

L
*

0
>

/

T

J

0

(

*

-

M

I

V

S

0

0

'

'

H

H

H

)

I
1

D

V

H

H

H

0

I
H

3

2

( l d O d O S ) V ( ;. : ‘ 9 ) J l l d l
(

v

n

u

=

i
(

t
I

4

t

)

V

i

: h

v

4

I

(

2
V

)
4

i
9

v
)

w
0

d
1

j
I

y

j

v
M

Z

2


CC 5; II = 1, NC
50 CP(II) = CA(II)
  1 FLAG = FLAG + CR
  CC 13 I = 1, NC, RP
  R(I) = SIN(FLCAT(1) * .38)
  RANG(XC, YC) = ANP = 2. - ANP
  JJ = 13, *NS = R(I)
  13 R(I) = FLCAT(JJ) / 13. *NS
  WRITE(4, 60) NC
50 FORMAT(I5, 39X, 4L6/40; GREEN CF; DATA 15)
  CC 61 I = 1, 8
  61 WRITE(6, 62) I, AA(I)
  62 FORMAT(2F3.1, 31 = F7.4)
  WRITE(6, 63) AA(5)
  63 FORMAT(15X, GAIA = F7.4)
  WRITE(6, 64) AA(1.)
  64 FORMAT(13X, CEAC TIME = F7.4)
  WRITE(6, 62) (R(I), BETA(1), I = 1, NC, NS)
  2 FORMAT(1X, 2F22.6)
  RETURN
END

SUBROUTINE ASSUM(IICPT)
    COMMON A(2, ), BET, BETA(2), DT, TOPS, CI (6, I6, 5), M( 7), X(13, 14),
    1 A, B, BETA(13, 13), MUC, MUC(13), CC(13), PHI, XA(13, 14)
    2 COMMON NCR, IPHAT, NT, NCRP1, NCRP2, NCRP1, NCRP2
12 FORMAT(7F13.5)
1C FORMAT(25B5, PHI for assumed solution FIC 5)
  IAPI = 1.
  CALL EPPRAP(41)
  IF (IAPI .NE. 1) CC 1C 2C.
IAP1 = 2
L = 1.
PH1 = 3.
C
A = ASSUREC PARAMETERS
C
THE PARAMETERS ARE READ IN CONSECUTIVE FIELDS WITH THE LAST
C
BEING THE GAIN
1F(IFF1, EC. 1) REAL(5, 12)(A(I), I = 1, NCRO + 1)
CC 2 I = 1,acre
2 C(I, 1) = 8.
CB = 9.
NCROF1 = NCROS + 1
CC 3 I = 1, NOBSF1
FLAG = R(I)
CR = (R(I) + R(I))/FLCSTR(NT)
CC 1 LL = 1, NT
5C CB = C(NCRO, L)
CC 4 II = 1, NCRO
4 CR = CB + CT * (-A(I) * C(I, L))
C(NCRO, L+1) = CE + A(NCROPI)*FLAG*DT
1F(NCRO , EC. 1) CC TO 7
CC 8 II = 1, NCROF1
KK = II + 1
8 C(II, L+1) = C(II, L) + CE*C(KK, L)
7 1F (L .NE. 1) CC TO 6
5 FORMAT(4H2, L, 5X, 2HC, 6X, 2HC3, 8X, 2HC4, 8X, 2HC5, 8X, 2HC6, PX,
12HC7, PX, 2HC8, 8X, 2HC9, 8X, 3HC13)
6 1F(IFNNT .EC. 1) WRITE(6,9)L, (C(P,L), M=1,NCRO)
9 FORMAT(15, L1F14.4)
FLAG = FLAG + DP
1 L = L + 1
LL = NT * (I - 1) + 1
PH1 = PH1 + (PE1A(I) - C(1,LL))*E=2
3 CONTINUE!
SUBROUTINE SOLN(IPT)
C
SUBROUTINE SOLUTION GENERATES THE FUNDAMENTAL SOLUTIONS
C
COMMON R(2,:), ETA(7,:), D, TCS(6,1003), A(7), X(13,14),
1 MAT(13,13), BVEC(13), C(13), PHI, XA(13,14)
C
COMMON NCPS, IFMAT, NI, NCRP, NCRP1, NCRP2, NCR, NCRP1, NCRP2
L = 1
CC 2 J = 1, NCRP1
CC 1 J = 1, NCRP2
XA(I,J) = .
1 X(I,J) = 0.
XA(I,1) = 0.
2 X(I,1) = 1.
CC 8 J = 1, NCPS
IF(IFMAT .NE. 1) GO TO 11
WRITE(6,60) J
60 FORMAT(12H_..OBSERVATION..15)
CC 50 K = 1, NCRP1
50 WRITE(6,61) (X(K,N), N = 1, NCRP2)
60 FORMAT(1X,7F17.5)
11 CALL MTRX(J)
16 CR = (R(J+1) - R(J))/FLCAT(N)
RR = R(J)
CC 3 K = 1, AT
CC 4 I = 1, NCRP2
CC 5 NW = 1, NCRP1
5 X(KK,1) = X(KK,1) + CT = XA(IFMAT+1,1)
CC 6 KK = 1, NCR
II = NCRP + KK
$X(MCRC, I) = X(MCRC, I) + DT \times \left( (-n(R)) \times XA(RK, I) \right) \times \left( (-C(I) \times L) \times XA(I, I) \right) \times \left( \left( \pm R \times XA(MCRP1, I) \right) \right)

IF(I \cdot FC, NCRP2) CC TO 15

CC = 4

15 CC 18 II = 1, NCRP

18 X(MCRC, I) = X(MCRC, I) + DT \times \left( A(I) \times C(I, L) \right)

4 CONTINUE

CC 12 II = 1, NCRP1

CC 12 RK = 1, NCRP2

13 XA(I, KK) = X(I, KK)

L = L + 1

3 RR = RR + DR

IF(1CFT.FC.1) CC TO 8

IF(J . FC. 1) PHI = L

K = N1 * (J-1) + 1

PHI = PHI + (BETA(J) - C(I, I)) * X

8 CONTINUE

RETURN

END

SUBROUTINE MTRX(J)

SUBROUTINE MATRX USES IFF INFORMATION GENERATED IN SOLVER:

TO FORM A SET OF ALGEBRAIC EQUATIONS WHICH CORRESPOND TO A

LEAST SQUARES CRITERION

CCMCA R(2,2), BETA(2,3), DT, ICBS, CI(6,13,3), AT(7), X(13,14),

1AMAT(13,13), BVEC(13), CC(13), PHI, XA(13,14)

CCMCA MCPS, IPMT, AI, MRPD, MRCPS1, MRDP1, MCR, NCR, MCRP1, NCRP2

IF(J . AF. 1) CC TO 1

CC 2 I = 1, NCRP1

BVEC(1) = 0.

CC 2 K = 1, NCRP1

2 AMAT(1, K) = 0.

1 CC 3 I = 1, NCRP1

BVEC(1) = BVEC(1) + (BETA(J) - X(1, MCRP2)) \times X(1, 1)
CC 3 K = 1, NCRP1
3 AMAT(I,K) = AMAT(I,K) + X(I,1) * X(I,K)
RETURN
END.

SUBROUTINE GALS(N)

SUBROUTINE GAUSS SOLVES THE EQUATIONS GENERATED IN MATRIX FOR
THE NEW SET OF PARAMETERS
RON INTERCHANGE IS USED
COMMON R(27), BETA(27), DI, T0CS, CJ(6,17), A(7), X(13,14),
1AMAT(13,13), BVEC(13), CJ(13), PHI, XA(13,14)
COMMON NCRS, IPRMT, M1, NCRD, NCRP1, NCRP2, NCR, NCRP1, NCRP2
DIMENSION AA(13,13), B(13), G(13)
EQUIVALENCE (AMAT(1,1), AA(1,1)), (B(1), BVEC(1)), (C(1), G(1))
IPRMT = 1
IF(IPRMT .NE. 1) CC 10 1
WRITE(6,81) (AA(I,J), J = 1, NCRP1), I = 1, NCRP1
80 FORMAT(3I8:\AMAT/(1X,5F17.9))
WRITE(6,81) (B(I), I = 1, NCRP1)
81 FORMAT(3I8:\BVEC/(1X, 7F17.8))

10 AN = N - 1
CC 5 I = 1, AN
J = N+1 - I
L = J - 1
CC 21 II = 1,L
IF(ABS(AA(J,J)) .GE. ABS(AA(II,J))) CC 10 21
CC 21 LL = 1, A
AS = AA(J,LL)
AA(J,LL) = AA(II,LL)
21 AA(II,LL) = AS
AS = B(II)
B(II) = AS
B(J) = AS
2C CONTINUE
CC 6 K = 1,L
FACT = 8A(K,J) /FA(J,J)
CC 7 M = 1,J
7 AA(K,P) = AA(K,M) - FACT * AA(J,P)
6 R(K) = R(K) - FACT * R(J)
5 CONTINUE
G(1) = B(1) /AA(1,1)
CC 11 I = 2,N
SUM = B(1)
5 = I - 1
CC 9 J = 1,P
9 SUM = SUM -AA(I,J) * G(J)
11 G(I) = SUM /AA(I,I)
IF(IFRNT .NE. 1) CC TO 12
WRITE(6,92)(G(I), I = 1,NCRP1)
82 FORMAT(3HCC/(1X, 7E17.8))
IFRNT = 3
12 RETURN
END

C
INPUT DATA
2 52
2 2.5 2.5
199. 199. 199. 199. 199. 199. 199. 199.
100. 100. 100. 100. 100. 100. 100. 100.
OPTIMUM GRADIENT
SUBROUTINE OPCRC(XL,U,F,PHI,ALX)
C OPTIMIZATION BY GRADIENT SEARCH
C P VECTOR IS PARAMETER VECTOR
C U VECTOR CONTAINS UPPER BOUND ON PARAMETER VECTOR
C XL VECTOR CONTAINS LOWER BOUND ON PARAMETER VECTOR
C
PREC = SUBROUTINE TO EVALUATE THE CRITERION FUNCTION
C GPHI=GRADIENT VECTOR OF CRITERION FUNCTION
C WP VECTOR CONTAINS CHANGES IN P VECTOR
C W VECTOR IS WEIGHTING FACTOR FOR CRITERION FUNCTION
C ASSUME ALL INTERVALS ARE WEIGHTED EQUALLY, WITH WEIGHT 1
C NOTE--IF UPPER AND LOWER BOUND ON PARAMETER ARE EQUAL THEN
C PARAMETER IS KEPT CONSTANT.
C NOTE--IF NOT INITIALIZED PARAMETERS OUTSIDE BOUNDS
C DIMENSION P(10),P1(10),P2(10),P3(10),DP(10),DPP(10),PM(10),GPHI(10)
C ID = 1
TLPS=0.501
C
C READ IN CONSTANTS
C
STC=1.
N1=-1
N=0
10 N1=N1+1
IF(N1.EQ.50) RETURN
20 CALL PREC(NLX,F,PHI)
ALX=2
IF(N1.EQ.1) GOTO1111
CC 5 J = 1, APARAM
5 IF (ABS(P(J) - C.) .GT. 1.E-1 ) GO TO 1111
GO TO 1
1111 STC=PHI
C
**CALCULATION OF GRADIENT AND MAGNITUDE OF GRADIENT SQUARED**

```fortran
CC 21 J=1,NPARM
21 P2(I)=P(I)
XPAG2=0.
CC 22 I=1,NPARM
DELT=ABS(CELI(I)+P(I))
IF(DELT.LT.PUTCCC) DELT=PUTCCC
P2(I)=P(I)+DELT
IF(P2(I).GT.PU(I))CC TO 22
IF(P2(I).GT.XL(I))CALL FC 22
CALL FRCCK(NLX,P2,PF11)
GPHI(I)=(PHI(I)-PI(I))/DELT
XPAG2=XPAG2+GPHI(I)**2
22 P2(I)=P(I)
GCTC(23,24),IF
23 WRITE(C,637) XI,PHI(I),P(I),GPHI(I),I=1,NPARM
WRITE(C,6,62) KJK
62 FORMAT(41F.,G., G. CRITERION FUNCTION EVALUATIONS = 115)
24 CALL SSATCH(2,ITSS)
CC TC(25,26),ITSS
25 STOP

**CALCULATION OF OPTIMUM STEP SIZE**

```fortran
CC 3E I=1,NPARM
CP(I)=-PHI*GPHI(I)/(2.*X+XPAG2)
3E P2(I)=P(I)+CP(I)
CALL FRES(LX,P2,P1)
CALL FRCCK(NLX,P2,PF12)
4E CC 5E I=1,NPARM
CP(I)=-DT(I)
5E P1(I)=P(I)+CP(I)
CALL PARRD(U,XL,P1)
```
CALL FRCG(NLX,P1,P11)
IF(P11-PF12.LE.0.)GO TO 171
IF(PF12-PF11.LT.0.)GO TO 63
P12=P11
K=N+1
GO TO 4C
60 GO TO 252
70 IF(P11-PF12.LE.0.)GO TO 140
K=P2(I)
50 P12=P(1)+K
CALL FRCG(NLX,P12,P2)
PHI1=PHI2
CALL FRCG(NLX,P1,P12)
K=N-1
IF(PH11-PH12.LE.0.)GO TO 140
IF(PN.LE.3)GO TO 130
GO TO 120
120 IF(K1=1, .NOT.NPARN)
CP(I)=2.*NP(I)
GO TO 80
130 PHI1=PHI12
140 CC 15C I=1, .NOT.NPARN
150 P(I)=P(I)+NP(I)
CALL FRCG(U,XL,P)
GO TO 180
160 K=N+1
GO TO 170
170 IF(K1=1, .NOT.NPARN)
CP(I)=.5*CP(I)
GO TO 210
171 K=N+1
GO TO 172
172 P(I)=P(I)+CP(I)
CALL FARBCD(U, XL, P)
CALL FRCG(XL, P, PHII)
IF(PHII-PHI, LE., .) GOTO 130

180 PHII = PHII
PHII = PHII
GOTO 171

190 IF(PHII-PHI, GT. C.) GOTO 140
CC 230 I = 1, KPARMT

200 CP(I) = 2.*OP(I)
CC 220 I = 1, KPARMT
CPM(I) = .75*CP(I)*(PHI2-5.*PHII+4.*PHII)/((PHI2-3.*PHII+2.*PHII)

220 PM(I) = P(I) + CPM(I)
CALL FARBCD(U, XL, P)
CALL FRCG(XL, P, PHII)
IF(PHII-PHI, GE., .) GOTO 240

230 P(I) = F(I) + 2.*CP(I)
CC 230 I = 1, KPARMT

240 PHI = PHI
GOTO 120

250 PHI = PHI2
GOTO 140

260 P(I) = F(I) + 2.*CP(I)
CALL FARBCD(U, XL, P)
GOTO 110

996 FORMAT(13/(3F15.5))
997 FORMAT(15H ITERATION NO. 13, 5x) CRITERION FUNCTION E26.5/
1 1.H COOP PONENTS X5 PARMETER EX8I GRADIENT/(I*X5, F14.5, E13.5))
996 FORMAT(1X, 70H
1)
995 FORMAT(5P14.5CP)
994 FORMAT(/)
992 FORMAT(//64X,221PCLADS ON PARAMETERS//46X,EL11.5,IX,14PG.E. KG L. CL 0162
1E.,1X,EL11.5/46X,EL11.5,1X,14PG.E. TR L.E.,1X,EL11.5 )
1 RETURN
END
SUBROUTINE PAR2CC(U, XL, PT)
COMMON L, XP(201), YP(201), XPARM, KJK
C THIS SUBROUTINE KEEPS PARAMETERS WITHIN THEIR SPECIFIED BOUNDS.
DIMENSION U(L), XL(L), PT(L)
DO 10 I=1, XPARM
IF (PT(I) .GT. U(I)) PT(I) = U(I)
IF (PT(I) .LT. XL(I)) PT(I) = XL(I)
10 CONTINUE
RETURN
END
GRADIENT PARTAN
C OPTIMIZATION BY GRADIENT DESCENT
C
C PROCEDURE TO EVALUATE CRITERION FUNCTION
C DIMENSION XL(13),U(13),P(13),P1(13),P2(13),P3(13),CPHI(I)
A1 = -1
AGCC = 0
AGRD = 0
JRR = 1
XMAE2 = 0
NLX = 0
REALS59, NPARNT, (XL(I),U(I),P(I),NPAR(I),I = 1,NPARNT)
999 FCV(13/(3Fl...II))
C EVALUATION OF THE CRITERION FUNCTION AT THE INITIAL POINT
CALL PROC(NLX,P,P1)
 1 CC 10 I = 1,NPARNT
10 P1(I) = P(I)
PHI1 = PHI1
NLX = 1
KLM = 3
GC TC 102
2 CC 20 I = 1,NPARNT
20 P2(I) = P(I)
PHI2 = PHI2
JRR = 2
NLX = 2
KLM = 3
GC TC 103
3 CC 30 I = 1,NPARNT
30 P3(I) = P(I)
PHI3 = PHI3
IF(ALX .LT. 0) GC TC 230
XMAE2 = 0
GC 11 I = 1,NPARNT
CPHI(I) = P2(I) - P3(I)
11 XPAG2 = X*AG2 + GPHI(1) * #2
  KLM = 2
  NKL = 3
  CC TO 100
200 X*AG2 = #2
  CC 20 I = 1, NPARMT
  GPHI(I) = P1(I) - P2(I)
22 X*AG2 = X*AG2 + GPHI(1) * #2
  KLM = 1
  ALX = 3
100 NR = NORT
  IF(NLX .NE. 3) NR = NACC
  CALL DARTAN(XL, L, F, PHI, NLX, NI, NR, GPHI, XPAG2)
  IF(NI .GE. 58) CC TO 8
  CC 5 J = 1, NPARMT
  5 IF(APS(P(J) - APS(J + 1)) .GT. .CC1) CC TO 1111
     CC TO 8
1111 IF(JPI .GE. 1) CC TO 2
  IF(KLP .GE. 1) CC TO 1
  IF(KLP .GE. 2) CC TO 2
  IF(KLP .GE. 3) CC TO 3
  8 WRITE(6,31) PHI
  31 IFORMAT(30000) OPTIMAL VALUES FOR PARAMETERS/7LUXPFI = F10.5
     CC 5 I = 1, NPARMT
     L = I + 1
  9 WRITE(6,31) L, P(I)
  31 IFORMAT(2H8, 11, 2E = E11.5)
  STOP
END

SUBCUTLINE DARTAN(XL, U, F, PHI, NLX, NI, N, GPHI, XMAG2)
C P VECTOR IS PARAMETER VECTOR
C U VECTOR CONTAINS UPPER BOUNDS ON PARAMETER VECTOR
C XL VECTOR CONTAINS LOWER BOUND ON PARAMETER VECTOR
C PHI=GRADIENT VECTOR OF CRITERION FUNCTION
C EP VECTOR CONTAINS CHANGES IN P VECTOR
C W VECTOR IS WEIGHTING FACTOR FOR CRITERION FUNCTION
C ASSUME ALL INTERVALS ARE WEIGHTED EQUALLY, WITH WEIGHT 1
C NCTE--IF UPPER AND LOWER BOUNDS ON PARAMETER ARE EQUAL THEN
C PARAMETER IS_kept_constant.
C NCTE--DO NOT INITIALIZE PARAMETERS OUTSIDE_BOUNDS
C DIMENSION P(10),P1(10),P2(10),PX(10),PL(10),EP(10),PE(10),GPHI(10),W(LX)
CC LCI(0,00,0,01), NLX
850 TEPS=.01
C READ IN CONSTANTS
C STC=1.
C
2G CALL FRCG(NLX,P,PHI)
IF (STC.NE.1) GOTO (1111
1111 STD=PHI
C
C CALCULATION OF GRADIENT AND MAGNITUDE OF GRADIENT SQUARED
C
CC 21 I=1, NPAR+1
21 P2(I)=P(I)
XPAC2=0.
CC 22 I=1, NPAR+1
DELTA=APS(.BTC1-P(I))
IF (DELTA.LT.0.) DELTA=0.
DELTA
P2(I)=P(I)+DELTA
IF (P2(I).GT.PL(I)) GOTO 21
IF (P2(I).LT.XL(I)) GOTO 22
CALL FRCG(NLX,P2,PHI)
GPHI(I)=(PHI-PHI)/DELTA
XPAC2=XPAC2+GPHI(I)*=2
22  P2(I)=P(I)
23  N1 = N1 + 1
24  CALL SS\$TCH(2, 11SS)
     GC TC(2, 2a), 11SS
25  STOP

C
C   CALCULATION OF OPTIMUM STEP SIZE
C
26  DC 3C  I=1,NPAR\$T
     CP(I)=-PHI*GPHI(I)/(2.*SH*PHI2)
30  P2(I)=P(I)+CP(I)
     CALL PAR\$GD(U, XL, P2)
     CALL FRCC(MLX, P2, PHI2)
40  DC 5C  I=1,NPAR\$T
     CP(I)=.5*DP(I)
50  P1(I)=P(I)+CP(I)
     CALL PAR\$GD(U, XL, P1)
     CALL FRCC(MLX, P1, PHI1)
     IF(PHI-PHI2.LT.1.) GO TO 171
     IF(PHI2-PHI1.LT.1.) GO TO 63
     PHI2=PHI1
     N=N+1
     GC TC 41
60  IF(N.LE.0) GC TC 252
     DC 7C  I=1,NPAR\$T
    70  CP(I)=4.*CP(I)
80  CC 9C  I=1,NPAR\$T
90  P2(I)=P(I)+CP(I)
     CALL PAR\$GD(U, XL, P2)
     PHI2=PHI1
     PHI1=PHI2
     CALL FRCC(MLX, P2, PHI2)
     N=N-1
     IF(PHI-PHI2.LT.1.) GC TC 14.
IF (K, LE, S) GC TC 12C
CC 12C I = 1, NPARRT
12C DP(I) = 2 * DP(I)
GC TC 20C
13C PHI = PHI +
14C CC 15C I = 1, NPARRT
15C P(I) = F(I) + DP(I)
CALL PARBCD(U, XL, P)
RETURN
16C K = K + 1
CC 17C I = 1, NPARRT
17C CP(I) = .5 * DP(I)
GC TC 21C
171 K = K + 1
CC 172 I = 1, NPARRT
172 CP(I) = .5 * DP(I)
CC 173 PHI = PHI +
CC 174 CP(I)
CALL PARBCD(U, XL, P)
CALL FOCC(NLX, P, PHI)
IF (PHI1 - PHI2) GC TC 19C
19C PHI2 = PHI1
PHI1 = PHI2
CC TC 171
19C IF (PHI1 - PHI2) GC TC 19C
CC 20C I = 1, NPARRT
20C CP(I) = 2 * DP(I)
CC 22C I = 1, NPARRT
22C CP(I) = .75 * CP(I) * (PHI2 - 5 * PHI1 + 4 * PHI2) / (PHI2 - 3 * PHI1 + 2 * PHI2)
22C PHI2 = PHI1 + CP(I)
CALL PARBCD(U, XL, P)
CALL FOCC(NLX, P, PHI)
IF (PHI1 - PHI2) GC TC 24C
CC 23C I = 1, NPARRT
23C PHI = PHI2 + CP(I)
CALL FARECD(U,XL,P)

RETURN

24c PHI=FI11

CC 1C 14:

25c PHI=FI12

DC 26c 1=1,NPARM1

26c P(I)=P(I)+2.*CP(I)

CALL FARECD(U,XL,P)

RETURN

995 FORMAT(13/(3F16.7))

997 FORMAT(15F5.5,B13)CRITERION FUNCTION E26.5/

1 15F COMPONENT X15 PARAMETER X15 GRADIENT/(1X15,F18.5,E15.5))

996 FORMAT(1X,74F

1)

995 FORMAT(5H14CP)

994 FORMAT(//)

993 FORMAT(1X,54X,21F10.5 PARAMETERS//46X,F11.5,1X,14H.G.E. KC L.

1E.,1X,E11.5/46X,E11.5,1X,14H.G.E. TR L.E.,1X,E11.5 )

1 RETURN

END

SUBROUTINE FARECD(U,XL,PT)

THIS SUBROUTINE KEYS PARAMETERS WITHIN THEIR SPECIFIED BOUNDS.

DIMENSION U(1C),XL(1C),PT(1H)

CC 1C 1=1,NPARM1

IF(PT(I).GT.U(I))PI(I)=U(I)

IF(PT(I).LT.XL(I))PI(I)=XL(I)

1C CONTINUE

RETURN

END
DAVIDON'S METHOD
SUBROUTINE DAVIEC(EPS, C, NP, P, PH, ELTA, IDRA, PMIN)

CAVIECK OPTIMIZATION

CA VARIABLE METRIC METHOD FOR MINIMIZATION

C BLOCK NUMBERS REFER TO PAPER VARIABLE METRIC METHOD FOR

C MINIMIZATION BY WILLIAM C. DAVIDSON

C REAC]

C THE FUNCTION OF THIS SECTION IS TO ESTABLISH A DIRECTION ALONG
C WHICH TO SEARCH FOR A RELATIVE MINIMUM, AND TO BOX OFF AN INTERVAL
C IN THIS DIRECTION WITHIN WHICH A RELATIVE MINIMUM IS LOCATED.
C IN ADDITION, THE CRITERION FOR TERMINATING THE ITERATIVE
C PROCEDURE IS EVALUATED.

C DIMENSION H(12,12),C(15),P(1.),S(1.),CP(1.),PP(1.),SP(2.),

1TLC(1.),CGS(1.),CR(1.),SIGMA(12)

110 FORMAT(15F10.5) FOR ASSUMED Solution.
111 FORMAT(11,15X,3HPR,2X,6HOLINV.)
112 FORMAT(15X,METRIC MATRIX)
113 FORMAT(11,15F15.5)
114 FORMAT(11,15F11.5)
115 FORMAT(14X,ITERATION NO.,5X,1277H;PHI = E18.8)
120 FORMAT(15X,OPTIMAL VALUES)
121 FORMAT(20X,DETERMINANT CF H = E15.8)

CALL CERF(C,PHI,P,NP)

C DPHI = PHI

WRITE(6,121) PHI

WRITE(6,125)

IF J = 1,NP

1 WRITE(6,126) J,F(J),C(J)

WRITE(6,126)
CC  2  J = 1,AP
WRITE(K,111) J, (P(J,JJ),JJ = 1,AP)
KX = 1
3 I = 1
4 EFF = EP * PHI
RLAM = 2.
5 CALL MULT(M,G,S,-1.,NP,1)
CALL INPRCD(S,C,CS,AP)

S IS THE DIRECTION OF THE STEP
GS IS THE COMPONENT OF THE GRADIENT IN THE DIRECTION OF THE STEP.
-1/2 GS IS THE EXPECTED IMPROVEMENT IN THE FUNCTION. IF -1/2 GS
IS LESS THAN THE MINIMUM IMPROVEMENT OF PHI READ IN, THE END GAME
IS INITIATED.

IF(GS*EFF .GT. 2.) GC TO 2C
WLAM = RLAM

CHECK STEP SIZE

CLAM = -2.* (PHI - C) / GS
IF(CLAM .LT. RLAM) WLAM = CLAM
SGS = -GS

TAKE STEP TWICE THE SIZE TO LOCATE PREDICTED MINIMUM OR
THE MAXIMUM STEP ALLOWED.

1G CC21C  J = 1,AP
21C EFF(J) = P(J) + WLAM * S(J)
CALL BOUND(PF,KXL)
IF(KXL .LE. 0) GC TO 11
WLAM = .5 * WLAM
GC TO 1C
EVALUATE FUNCTION AND GRADIENT AT NEW POINT

11 CALL CEROF(GP,PHI,EP,NP)
   CALL INPREOTS(GP,GSP,NP)                   BLOCK 12
C
C CHECK IF MINIMUM WITHIN STEP. IF YES GO TO AIM.
C
IF(GSF .GT. 0.) GC TO 21
IF(PHI .GT. PHI) GC TO 21                        BLOCK 13

C STEP TO SMALL SC CALBLE STEP AND START OVER.
C
IF(WLAP .EQ. RLAP) GC TO 117
   GC 16 11 = 1,AP
   GC 16 J = 1,AP
   18 F(I,I,J) = F(I,I,J) + S(I1) * S(J) / SCS
   DELTA = 2. * CELTA
   GC TC 17
117 RLAP = 2. * RLAP
   17 GC217 J = 1,AP
   P(J) = FP(J)
217 G(J) = GP(J)
   PHI = PHI
   GC TC 5
   20 GC220 = 1,AP
220 P(J) = P(J) + S(J)
   GC TC 67

AIM:

ESTIMATES LOCATION OF THE RELATIVE MINIMUM WITHIN INTERVAL
SELECT BY REACY. ALSO A COMPARISON IS MADE OF THE
IMPROVEMENT EXPECTED BY GOING TO THIS MINIMUM WITH THAT FROM
A STEP PERPENDICULAR TO THIS DIRECTION..
INTERPOLATE FOR THE LOCATION OF THE MINIMUM BY CHOOSING THE
SPECTRUM CURVE SATISFYING THE BOUNDARY CONDITIONS AT P AND PP.

21 Z = 3 * (PH1 - PH1P) / WLA M + GS + GSP
C = AES(Z) * SCRT(1. - (GS * CSF/Z))
A = (GSP + C - Z) / (GSP - GS + 2. * Q)
ALPHA = WLA M * (1. - A)
PHIC = PHIP - TC

CHECK STEP PERPENDICULAR TO PREDICTED MINIMUM.

CALL MULT(H, CP, SP, -1, NP, 1)
DC 26 J = 1, NP
26 TU(J) = SP(J) + (GSP * S(J)) / SGS
CALL INFRCD(TU, CP, CT, NP)
IF(Z. * TC + GTP . GT. C.) GO TO 28
IF(.5 * GTP + PH1 - P . GT. C.) GO TO 30

STEP IS BEING TAKEN IN PERPENDICULAR DIRECTION.

DC 30 J = 1, NP
30 TU(J) = TU(J) + FP(J)
CALL DERF(SH, PHIE, TL, NP)
IF(PHIP - PHIE . GT. C.) GO TO 38

STEP IN PERPENDICULAR DIRECTION IS BEING TAKEN.

DC 32 J = 1, NP
33 S(J) = TU(J) - F(J)
SGS = -CTP + SGS + WLA M * 2
WLA M = 1.
DC 35 J = 1, NP
35 GUS(J) = G9(J) - G(J)
CALL INPROC(S,CLS,CSS,GP)
TES = GSS * EPP
IF(TES) 36,37,38
36 IPX = 2
CC TC 5:
37 IPX = 1
CC TC 5:
38 CC 238 J = 1,NP
SIGMA(J) = ALPFA * S(J)
238 T(U(J)) = A * P(J) + (1. - A) * PP(J)

C**********************************************************************
C THE PURPOSES OF THIS SECTION ARE TO EVALUATE THE FUNCTION AND ITS
C GRADIENT AT THE INTERPOLATED POINT AND TO DETERMINE IF THE LOCAL
C MINIMUM HAS BEEN SUFFICIENTLY WELL LOCATED.
C**********************************************************************
CALL CERF(CE,PHI,PL,GP)
CALL INPROC(S,CLS,CSP,GP)
F = PHI
IF(PHI > LT, PHI) F = PHI
IF(F - PHI > EPP .LT. 1.) CC TC 46
TC = GSP * (A/(1. - A)) - (1. - A)/A)
IF(ABS(TC) < C .LT. C.) GO TO 45
CSS = TC + 2. * C
CC 45 J = 1,NP
45 CUS(J) = (GR(J) - C(J)) = (A/(1. - A)) + (GP(J) - GR(J)) * (1. - A)/A
IPX = 1
CC TC 5.
46 IF(PHI > DT) GO TO 48
H1NAP = H1NAP * (1. - A)
PHI = PHI
CSP = GSB
CC 47 J = 1,NP
47 \( CP(J) = GP(J) \)

48 \( WLAP = A \times WLAP \)

\( P(J) = P(J0) \)

\( GS = GSP \)

\( CC24E J = 1, NP \)

\( P(J) = TU(J) \)

49 \( GSS = 2 \times G \)

\( IPX = 3 \)

Block 48

The purpose of this section is to modify the metric \( H \) on the basis of information obtained about the function along the direction \( S \).

50 \( CC23C J = 1, NP \)

51 \( P(J) = TU(J) \)

\( CC TC (51, 57, 59), 1 \times \)

52 \( CALL \{ MULTC, GLS, TLJ, N, M, 1 \} \)

\( CALL \{ MPRDQ(TC, CUS, TC, NP) \} \)

\( IF(TC - GSS * 2 / GSS - EPP, LT, 0) \) GO TO 55

\( CC54 II = 1, NP \)

\( CC54 J = 1, NP \)

53 \( H(IJ, J) = H(IJ, J) - T(J) \times TL(J) / TC \)

\( DELTA = DELTA * WLAP * GSS / TC \)

\( CC56 II = 1, NP \)

\( CC56 J = 1, NP \)

54 \( H(IJ, J) = H(IJ, J) + (WLAP/GSS) * S(I) * S(J) \)

55 \( PFI = PF10 \)

\( CC29E J = 1, NP \)

Block 49
258 G(J) = CE(J)
    IF(IPFM, .EQ., 0) CC 10158
    WRITE(6,115) I,PHI
    WRITE(6,105)
    DC257 J = 1,AP
257 WRITE(6,115) J,F(J),G(J)
    WRITE(6,106)
    CC 357 J = 1,AP
357 WRITE(6,111) J, (F(J,JJ), JJ = 1,AP)
    WRITE(6,130) CELTA
158 IF(KX .NE. KX + 1) GO TO 59
    CC 457 IL = 1,AP
    CC 457 J = 1,AP
457 P(IL,IL) = 1.
    CELTA = 1.0
    KX = 0
58 I = I + 1
    IF(I .EQ. 21) RETURN
    KX = KX + 1
    IF(PHI + PMIN .CF. CLDPHI) CC TC 67
    CLDPHI = PHI
    CC 72 J = 1,AP
    IF(AES(S(I,J)) .CF. PFIN) GO TO 4
75 IF(AES(SIGMA(J)) .CF. ACOM) CC TC 4
    CC TC 67
59 CELTA = CELTA * WLM * SGS / CSS
    CC 65 II = 1,AP
    CC 66 J = 1,AP
60 H(II,J) = H(II,J) + S(II) * S(J) * WLM / (CSS * SGS)
    CC TC 57
67 WRITE(6,123)
    WRITE(6,115) I,PHI

BLOCK 57

BLOCK 58

BLOCK 59

BLOCK 60

BLOCK 67
SUBROUTINE BOUNDS(X, I)
DIMENSION X(I)
I = 6
IF(X(1) .LT. 2.) GO TO 170
IF(X(2) .GT. X(3) + 1.) GO TO 173
IF(X(2) .LT. (-X(3)-1.)) GO TO 175
IF(X(3) .GT. 1.) GO TO 175
IF(X(5) .LT. 6.) GO TO 175
IF(X(5) .GT. (4.-X(6))/2.) GO TO 175
IF(X(6) .LT. 6.) GO TO 175
RETURN
170 I = 1
RETURN
END

SUBROUTINE CERT(G, PHI, P, NP).

THE FUNCTION OF THIS SUBROUTINE IS TO EVALUATE THE GRADIENT.
IF AN ANALYTICAL EXPRESSION IS AVAILABLE FOR THE DERIVATIVE IT
SHOULD BE USED IN THIS SUBROUTINE.

CALL FURC(PHI, P, NP)
CC 10  I = 1, NP
10 P2(I) = P(I)
CC 11  I = 1, NP
DELTA = ABS(.CDC11 * P(I))
IF(DELTA .LT. .CDC11) DELTA = .CDC11
13 P2(I) = P(I) + DELTA
12 CALL FUN(PHI1, P2, NP)
G(I) = (PHI1 - PHI) / DELTA
11 P2(I) = P(I)
RETURN
END

SUBROUTINE MULT(A, P, C, D, N, NA, NC)
COMMON XA(5C), YA(5C), N, KJK, GAIN
DIMENSION A(I, I), B(I), C(I)
CC I I = 1, NA
C(I) = 0.
CC 1 K = 1, NA
1 C(K) = C(I) + A(I, K) * B(K) * C
RETURN
END

SUBROUTINE IMPREC(X, Y, Z, L)
COMMON XA(5C), YA(5C), N, KJK, GAIN
DIMENSION X(I), Y(I)
Z = (..:.
CC 1 KA = 1, L
SUP = X(KA) * Y(KA)
41 Z = Z + SUP
RETURN
END

SUBROUTINE INPUT(FF, C, NP, P, P, DELTA, IPRT, PFIN)
COMMON XA(5C), YA(5C), N, KJK, GAIN
THIS PROGRAM READS IN NECESSARY INFORMATION FOR DAVIDCH

C DIMENSION P(I,J), H(I,J)
C REAC(I,J) EP, C, PMIN, NP, IPRAT
10 FORMAT(3F10.5,210)
C
EP = THE TIMES FRACTIONAL ACCURACY TO WHICH THE FUNCTION
IS TO BE MINIMIZED
C = A LIMITING VALUE FOR WHAT IS TO BE CONSIDERED AS A
REASONABLE MINIMUM VALUE OF THE FUNCTION. FOR LEAST SQUARES
PROBLEMS C CAN BE SET EQUAL TO ZERO.
C PMIN IF ALL PARAMETERS CHANCE BE LESS THAN THIS AMOUNT
OR IF ALL COMPONENTS OF THE DIRECTION ARE LESS THAN THIS THE
SEARCH IS TERMINATEC
NP = NUMBER OF PARAMETERS IN THE SYSTEM.
IPRAT = C. WILL ELIMINATE PRINTING AT EACH ITERATION.
C
REAC(I,J)(P(I), I = 1, NP)
15 FORMAT(9F1.5)
C
P = INITIAL GUESSES FOR ALL PARAMETERS.
C
CC 2C I = 1, NP
CC 2C J = 1, AP
C(I,J) = 0.
20 H(I,J) = 1.
C DETA = 1.
C
F- IS A NON-NEGATIVE SYMMETRIC MATRIX WHICH WILL BE USED AS A
METRIC IN THE SPACE OF THE VARIABLES. IF AN ESTIMATE OF THE
ACCURACIES OF THE INITIAL PARAMETERS ARE KNOWN, THEIR SQUARES SHOULD BE
USED FOR THE DIAGONAL MATRIX F REPLACING THE UNIT MATRIX. THE
BEST KNOWN VALUES OF F SHOULD BE USED IF CONTINUING A PROBLEM.

THE F USED IN THIS PROGRAM ASSURES THAT THE FIRST STEP TAKEN
IS IN THE DIRECTION OF STEEPEST DESCENT.

DELTA IS THE DETERMINANT OF F.

RETURN
END
CONJUGATE GRADIENTS
SUBROUTINE FMCC

PURPOSE

TO FIND A LOCAL MINIMUM OF A FUNCTION OF SEVERAL VARIABLES
BY THE METHOD OF CONJUGATE GRADIENTS

USAGE

CALL FMCC(FUNCT,N,X,F,G,EST,LPS,LIMIT,IER,F)

DESCRIPTION OF PARAMETERS

FUNCT - USER-WRITTEN SUBROUTINE CONCERNING THE FUNCTION TO
         BE MINIMIZED. IT MUST BE OF THE FORM
         SUBROUTINE FUNCTION(ARG,VAL,GRAD)
         AND MUST SERVE THE FOLLOWING PURPOSE
         FOR EACH N-DIMENSIONAL ARGUMENT VECTOR ARG,
         FUNCTION VALUE AND GRADIENT VECTOR MUST BE COMPUTED
         AND, ON RETURN, STORED IN VAL AND GRAD RESPECTIVELY

N - NUMBER OF VARIABLES

X - VECTOR OF DIMENSION N CONTAINING THE INITIAL
    ARGUMENT WHERE THE ITERATION STARTS, ON RETURN,
    X HOLDS THE ARGUMENT CORRESPONDING TO THE
    COMPUTED MINIMUM FUNCTION VALUE.

F - SINGLE VARIABLE CONTAINING THE MINIMUM FUNCTION
    VALUE ON RETURN, I.E. F=F(X).

G - VECTOR OF DIMENSION N CONTAINING THE GRADIENT
    VECTOR CORRESPONDING TO THE MINIMUM ON RETURN,
    I.E. G=G(X).

EST - IS AN ESTIMATE OF THE MINIMUM FUNCTION VALUE.

EPS - TEST VALUE REPRESENTING THE EXPECTED ABSOLUTE ERROR.
      A REASONABLE CHOICE IS 1.E-36, I.E.
      SOMEWHAT GREATER THAN 1.E-D, WHERE D IS THE
      NUMBER OF SIGNIFICANT DIGITS IN FLOATING POINT
      REPRESENTATION.

LIMIT - MAXIMUM NUMBER OF ITERATIONS.

IER - ERROR PARAMETER
      IER = 2 MEANS CONVERGENCE WAS OBTAINED
IER = 1 MEANS NO CONVERGENCE IN LIMIT ITERATIONS
IER = -1 MEANS ERRORS IN GRADIENT CALCULATION
IER = 2 MEANS LINEAR SEARCH TECHNIQUE INDICATES IT IS LIKELY THAT THERE EXISTS NO MINIMUM.

- WORKING STORAGE IN DIMENSION 2*N.

REMARKS
I) THE SUBROUTINE NAME REPLACING THE DUMMY ARGUMENT FUNCT
MUST BE DECLARED AS EXTERNAL IN THE CALLING PROGRAM.
NAMELY DERF EXTERNAL DERF
II) IER IS SET TO 2 IF, STEPPING IN ONE OF THE COMPUTED
DIRECTIONS, THE FUNCTION WILL NEVER INCREASE WITHIN
A TOLERABLE RANGE OF ARGUMENT.
IER = 2 MAY OCCUR ALSO IF THE INTERVAL WHERE F
INCREASES IS SMALL AND THE INITIAL ARGUMENT WAS
RELATIVELY FAR AWAY FROM THE MINIMUM SUCH THAT THE
MINIMUM WAS OVERLOOKED. THIS IS DUE TO THE SEARCH
TECHNIQUE WHICH DOUBLES THE STEPSIZE UNTIL A POINT
IS FOUND WHERE THE FUNCTION INCREASES.

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED FUNCT
SUBROUTINE PROC(F,PHI) MUST EVALUATE CRITERION FUNCTION FOR
VECTOR P AND RETURN THE VALUE IN PHI. SUBROUTINE DERF WILL
TAKE CARE OF THE GRADIENTS.
KJ = NUMBER OF FUNCTION EVALUATIONS; IF PROC PROVIDES FOR
COUNTING, AND KJ IS ENTERED INTO COMMON

METHOD
THE METHOD IS DESCRIBED IN THE FOLLOWING ARTICLE
R.FLEISCHER AND C.M.REEVES, FUNCTION MINIMIZATION BY
CONJUGATE GRADIENTS

SUBROUTINE PROC(FLACT,N,X,F,G,EST,EPS,LIMIT,IER,H)
DIMENSION CUNKY VARIABLES
DIMENSION X(1),C(1),H(1)

COMPUTE FUNCTION VALUE AND GRADIENT VECTOR FOR INITIAL ARGUMENT
IER=0
CALL FUNCT(N, X, F, C, DER)
C
RESET ITERATION COUNTER
KCOUNT = 0
N = N + 1
C
START ITERATION CYCLE FOR EVERY N+1 ITERATIONS
1 DC 43 II = 1, N
C
STEP ITERATION COUNTER AND SAVE FUNCTION VALUE
KCOUNT = KCOUNT + 1
WRITE(6, 61) KCOUNT
61 FORMAT(14HITERATION NO. II. )
WRITE(6, 61) (X(I), I = 1, N)
61 FORMAT(2E15.5)
WRITE(6, 02) F, KJ
62 FORMAT(7F, 5/28HNO. OF FUNCTION EVALUATIONS 112)
CLDF = F
C
COMPUTE SQUARE OF GRADIENT AND TERMINATE IF ZERO
GGRF = G.
DC2J = 1, N
2 GGRF = C GRF + 5(J) * G(J)
IF(GGRF)46, 6
C
EACH TIME THE ITERATION LOOP IS EXECUTED, THE FIRST STEP WILL
BE IN DIRECTION OF STEEPEST DESCENT.
3 IF(II - 1)4, 4, 6
4 DC5J = 1, N
5 H(J) = -G(J)
Ge. TC B
C
FURTHER DIRECTION VECTORS F WILL BE CHOSEN CORRESPONDING
C
to the conjugate gradient method.
6 AMBDA = GGRF/CLEG
CC7J = 1, N
7 H(J) = AMBDA * F(J) - G(J)
C
COMPUTE TESTVALUE FOR DIRECTIONAL VECTOR AND DIRECTIONAL
C
DERIVATIVE
8 CY = .
SAVE ARGUMENT VECTOR
F(K)=X(J)
FNRF=FNRF+ARF(H(J))

CY=CY+H(J)*G(J)
CHECK WHETHER FUNCTION WILL DECREASE STEPPING ALONG H AND
SKIP LINEAR SEARCH ROUTINE IF NOT
IF(CY1,1,42,42
COMPUTE SCALE FACTOR USED IN LINEAR SEARCH SUBROUTINE
SNRF=1./SNRF
SEARCH MINIMUM ALONG DIRECTION H
SEARCH ALONG H FOR POSITIVE DIRECTIONAL DERIVATIVE
FY=F
ALFA=2.*(EST-F)/LY
AMP=SNRF
USE ESTIMATE FOR STEPWISE ONLY IF IT IS POSITIVE AND LESS THAN
SNRF, OTHERWISE TAKE SNRF AS STEPSIZE.
IF(ALFA13,13,11
IF(ALFA-AMP)12,13,13
AMP=ALFA
ALFA=C.
SAVE FUNCTION AND DERIVATIVE VALUES FOR OLD ARGUMENT
FX=FY
CX=CY
STEP ARGUMENT ALONG H
CC151=1,N
X(I)=X(I)+AMP*AI(I)

COMPUTE FUNCTION VALUE AND GRADIENT FOR NEW ARGUMENT
CALL FUNCTION(X,X,F,G,FY)
FY=F
COMPUTE DIRECTIONAL DERIVATIVE DY FOR NEW ARGUMENT, TERMINATE
SEARCH, IF BY POSITIVE. IF BY IS ZERO THE MINIMUM IS FOUND.
DY =  
CC161 = 1, K
16 CY = CY + C(1) + F(1)
   IF (CY) 17, 33, 2
   C TERMINATE SEARCH ALSO IF THE FUNCTION VALUE INDICATES THAT
   C A MINIMUM HAS BEEN PASSED
   17 IF (FY - FX) 18, 27, 2:
   C REPEAT SEARCH AND DOUBLE STEP SIZE FOR FURTHER SEARCHES
   18 AMBDA = AMBDA + ALFA
   ALFA = AMBDA
   C TERMINATE IF THE CHANGE IN ARGUMENT GETS VERY LARGE
   C IF (ARR*AMBDA - 1.15) 14, 14, 19
   C LINEAR SEARCH TECHNIQUE INDICATES THAT NO MINIMUM EXISTS
   19 IER = 2
   RETURN
   C END OF SEARCH LOOP
   C INTERPOLATE CUBICALLY IN THE INTERVAL DEFINED BY THE SEARCH
   C ZERO AND COMPUTE THE ARGUMENT X FOR WHICH THE INTERPOLATION
   C POLYNOMIAL IS MINIMIZED.
   20 T = 0.
   21 IF (AMBDA) 22, 36, 22
   22 Z = 3.* (FX - FY) / AMBDA + CX + CY.
   ALFA = AMBDA (AFX) + ARS(0X) + ARS(CY))
   131 DALFA = Z / ALFA
   DALFA = DALFA + DALFA - CX / ALFA + CY / ALFA
   IF (CALFA) 23, 27, 27
   C RESTORE OLD VALUES OF FUNCTION AND ARGUMENTS
   23 DC24J = 1, N
   K = K + 1
   24 X(J) = F(K)
   CALL FUNCT(N, X, F, C, IER)
   C TEST FOR REPEATED FAILURE OF ITERATION
   25 IF (IER) 47, 26, 47
   26 IER = -1
CC TO 1

27 ALFA=SCRT(DALFA)
ALFA=(CY+x-7)*AMBCA/(DY+2.*W-LX)
CC281=1,

28 X(1)=X(1)+(T-ALFA)*F(1)
C TERMINATE, IF THE VALUE OF THE ACTUAL FUNCTION AT X IS LESS
C THAN THE FUNCTION VALUES AT THE INTERVAL ENDS. OTHERWISE REDUC
C THE INTERVAL BY CHOOSING ONE END POINT EQUAL TO X AND REPEAT
C THE INTERPOLATION, WHICH END-POINT IS CHOSEN DEPENDS ON THE
C VALUE OF THE FUNCTION AND ITS GRADIENT AT X.
CALL FUNCT(5,X,F,S,IER)
IF (F-FX) 22, 29, 3;
29 IF (F-FY) 38, 38, 3;
C COMPUTE DIRECTIONAL DERIVATIVE
35 DALFA=1.
30 CC311=1,
31 DALFA=DALFA+G(I)*F(I)
IF (DALFA) 32, 35, 35
32 IF (T-FX) 34, 33, 35
33 IF (CX-DALFA) 34, 28, 34
34 FX=F
35 CX=DALFA
36 T=ALFA
37 AMBDA=ALFA
CC TO 21
35 IF (FY-F) 37, 36, 37
36 IF (CY-DALFA) 37, 28, 37
37 FY = F
38 CY=DALFA
39 AMBDA = AMBDA - ALFA
CC TO 2.
C COMPUTE DIFFERENCE OF NEW AND OLD ARGUMENT VECTOR
38 T=5.
CC39J=1,
\[ K = J + A \]
\[ F(K) = X(J) - H(K) \]

39 \[ T = T + \text{APS}(I) \]

C TEST LENGTH OF DIFFERENCE VECTOR IF AT LEAST N+1 ITERATIONS
C HAVE BEEN EXECUTED, TERMINATE, IF LENGTH IS LESS THAN EPS

40 IF(KCNT - N1) \[41, 44, 45\]

42 IF(T - EPS) \[45, 47, 48\]

41 IF (CLS - F + EPS) \[48, 49, 50\]

42 CLOG = CLNR

41 TERMINATE, IF NUMBER OF ITERATIONS WOULD EXCEED LIMIT

43 IF(KCN1 - LIMIT) \[43, 44, 46\]

42 C END OF ITERATION CYCLE

43 C START NEXT ITERATION CYCLE

44 IC = 1

45 IF(CMNP - EPS) \[46, 47, 48\]

46 IF(CMNP - EPS) \[48, 49, 49\]

47 IF (IER) \[47, 49, 50\]

48 IF (IER) \[47, 49, 50\]

49 IER = -1

50 IC = 1

51 IER = 2

47 RETURN

END

SUBROUTINE DPERP(NF, F, PHI, G, IER)

C THE FUNCTION OF THIS SUBROUTINE IS TO EVALUATE THE GRADIENT.
IF AN ANALYTICAL EXPRESSION IS AVAILABLE FOR THE DERIVATIVE IT SHOULD BE USED IN THIS SUBROUTINE.

DIMENSION G(I), P(I), P2(I)

CALL FFCC(P, P(I))
CC 16 P2(I) = P(I)
CC 11 I = 1, NP
DELT = ABS(.5 + 1 = P(I))
IF(DELT .LT. .000001) DELTA = .000001
13 P2(I) = P(I) + DELTA
CALL FPCC(P2, P(I))
G(I) = (P2(I) - P(I)) / DELTA
11 P2(I) = P(I)
RETURN
END
FLETCHER POWELL
The text on the page is not legible due to the quality of the image. It appears to be a page from a book or a document, possibly containing technical or academic content. The text is not transcribed as it cannot be accurately read from the image provided.
IER = 1 MEANS NO CONVERGENCE IN LIMIT ITERATIONS
IER =-1 MEANS ERRORS IN GRADIENT CALCULATION
IER = 2 MEANS LINEAR SEARCH TECHNIQUE INDICATES
IT IS LIKELY THAT THERE EXISTS NO MINIMUM.
H = WORKING STORAGE OF DIMENSION \( k \times (n+5)/2 \).

REMARKS

1) THE SUBROUTINE NAME REPLACING THE DUMMY ARGUMENT FUNCTION
MUST BE DECLARED AS EXTERNAL IN THE CALLING PROGRAM.
NAMELY DERO, EXTERNAL DERF
2) IER IS SET TO 2 IF, STEPPING IN ONE OF THE COMPUTED
DIRECTIONS, THE FUNCTION WILL NEVER INCREASE WITHIN
A TOLERABLE RANGE OF ARGUMENT.
IER = 2 MAY OCCUR ALSO IF THE INTERVAL WHERE F
INCREASES IS SMALL AND THE INITIAL ARGUMENT WAS
RELATIVELY FAR AWAY FROM THE MINIMUM SUCH THAT THE
MINIMUM WAS OVERLEAPED. THIS IS DUE TO THE SEARCH
TECHNIQUE WHICH DOUBLES THE STEPSIZE UNTIL A POINT
IS FOUND WHERE THE FUNCTION INCREASES.

SUBROUTINES AND FUNCTION SUBPROGRAMS REQUIRED

FLACT
SUBROUTINE PREC(P, Phi) MUST EVALUATE CRITERION FUNCTION FOR
VECTOR P AND RETURN THE VALUE IN PHI. SUBROUTINE DERO WILL
TAKE CARE OF THE GRADIENTS.
KJK = NUMBER OF FUNCTION EVALUATIONS, IF PREC PROVIDES FOR
COUNTING, AND KJK IS ENTERED INTO COMFCH

METHOD
THE METHOD IS DESCRIBED IN THE FOLLOWING ARTICLE
R. FLETCHER AND R. J. D. POWELL, A RAPID DESCENT METHOD FOR
MINIMIZATION.

COMPUTER JOURNAL VOL. 6, ISS. 2, 1963, PP. 163-166.
SUBROUTINE FFPF(FLACT, M, X, F, C, EPS, LIMIT, IER, H)
DIMENSION COMMON VARIABLES
COMMON XA(5:5), YA(3..), NA, KJK, EPSV
DIMENSION H(1), x(1), C(1), XA(1..)
C      COMPUTE FUNCTION VALUE AND GRADIENT VECTOR FOR INITIAL ARGUMENT
C      CALL FUNCT(Y,N,f,C)
C      BLANK ITERATION CONTROL AND GENERATE IDENTITY MATRIX
     1 K=1,31
     2 CC4J=J,K
     3 F(K) = 1.
     4 NJ=J-J.
     5 IF(NJ)5,5,2
     6 CC3I=J,KJ
     7 KJ=K+L
     8 F(KL) = 0.
     9 K = KL + 1
C      START ITERATION LOOP
     10 K0UNT=K0UNT+1
     11 WRITE(6,61) K0UNT
     12 FORMAT(14I4,10X) ! (iteration no. 1)
     13 WRITE(6,61) (F(I), I = 1, J)
     14 FORMAT(11I8,10X) ! function evaluations 113)
C      SAVE FUNCTION VALUE, ARGUMENT VECTOR AND GRADIENT VECTOR
     15 CJCF=0
     16 CC6J=J,K
     17 F=CC6J
     18 I(K)=C(J)
     19 K=K+K
     20 LL = KJ+J
     21 I(JL) = X(J)
F(K) = X(J)

DETERMINE DIRECTION VECTOR H
K = J + N/2
T = C
CC8L = 1, K
T = T - C(L)*F(K)
IF(L - J) <= 7, 7
6 K = K + A - L
6 GC TC 8
7 K = K + 1
8 CONTINUE
9 F(J) = T
C CHECK WHETHER FUNCTION WILL DECREASE STEPPING ALONG H.
CY = C
FNRP = C
GNRP = C
C CALCULATE DIRECTIONAL DERIVATIVE AND TESTVALUES FOR DIRECTION
C VECTOR H AND GRADIENT VECTOR G.
DCl0J = 1, A
FNRP = FNRP + ABS(F(J))
GNRP = GNRP + ACS(G(J))
10 CY = CY + H(J)*G(J)
C REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF DIRECTIONAL
C DERIVATIVE APPEARS TO BE POSITIVE OR ZERO.
IF(CY) 11, 51, 51
C REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF DIRECTIONAL
C VECTOR H IS SMALL COMPARED TO GRADIENT VECTOR G.
11 IF(HNRN/GNRF - EPS) 51, 51, 12
C SEARCH MINIMUM ALONG DIRECTION H
C SEARCH ALONG H FOR POSITIVE DIRECTIONAL DERIVATIVE
12 FY = F
ALFA = 2.*((EST - F)/CY
AMBCA = 1.
C USE ESTIMATE FOR STEPSIZE ONLY IF IT IS POSITIVE AND LESS THAN
C 1. OTHERWISE TAKE I. AS STEPSIZE
IF(ALFA)15,15,1
13 IF(ALFA-APBEC)14,15,15
14 APBEC=ALFA
15 ALFA=C.
C SAVE FUNCTION AND DERIVATIVE VALUES FOR OLD ARGUMENT
16 FX=FY
C0=CY
CC 116 I = 1,N
116 XA(I) = X(I)
C STEP ARGUMENT ALONG H
117 CC171=1,N
17 X(I) = XA(I) + APBEC*H(I)
C CHECK STABILITY LIMITS
IF(X(I) .LT. .1.) GC TO 177
IF(X(2) .GT. X(2) + 1.) GO TO 177
IF(X(2) .LT. (-X(2)-1.)) GO TO 177
IF(X(2) .GT. 1.) GC TO 177
IF(X(5) .LT. 5.) GC TO 177
IF(X(5) .GT. (4.-X(6))/2.) GC TO 177
GC TO 177
C SYSTEM OUTSIDE OF LIMITS
177 APBEC = .40 * APBEC
ALFA = .40 * ALFA
GC TO 117
C COMPUTE FUNCTION VALUE AND GRADIENT FOR NEW ARGUMENT
177 CALL FUNCT(K,X,F,C)
FY=F
C COMPUTE DIRECTIONAL DERIVATIVE FY FOR NEW ARGUMENT. TERMINATE
C SEARCH, IF CY IS POSITIVE, IF CY IS ZERO THE MINIMUM IS FOUND.
C
CY=C.
CC181=1,N
18. CY=CY+G(I)*F(I)
IF(CY)15,36,22
C TERMINATE SEARCH ALSO IF THE FUNCTION VALUE INDICATES THAT
C A MINIMUM HAS BEEN PASSED.
19 IF (FY-FX) 27, 27, 28
C REPEAT SEARCH AND DOUBLE STEPSIZE FOR FURTHER SEARCHES
20 AMBCA = AMPDA + ALFA
ALFA = AMPDA
C END OF SEARCH LOOP
C TERMINATE IF THE CHANGE IN ARGUMENT GETS VERY LARGE
21 IF (FARK-AMBCA) 16, 16, 21
C LINEAR SEARCH TECHNIQUE INDICATES THAT NO MINIMUM EXISTS
C INTERPOLATE CUBICALLY IN THE INTERVAL DEFINED BY THE SEARCH
C ABOVE AND COMPUTE THE ARGUMENT X FOR WHICH THE INTERPOLATION
C POLYNOMIAL IS MINIMIZED.
22 I = 0.
23 AMBCA 24, 36, 24
24 z = 3, * (FX-FY)/AMBCA + CX + CY
ALFA = AMAX1 (ABS (Z), ABS (CX), ABS (CY))
CALFA = z/ALFA
CALFA = CALFA * DALFA - CX/ALFA * CY/ALFA
IF (CALFA) 51, 25, 26
25 h = ALFA * SQR (CALFA)
ALFA = (DY+W-Z)*AMBCA/(DY+2,W-CX)
C 26 I = I
27 X(I) = X(I) + (T-ALFA)*H(I).
C TERMINATE, IF THE VALUE OF THE ACTUAL FUNCTION AT X IS LESS
C THAN THE FUNCTION VALUES AT THE INTERVAL ENDS. OTHERWISE REDUCE
C THE INTERVAL BY CHOOSING ONE END-POINT EQUAL TO X AND REPEAT
C THE INTERPOLATION. WHICH END-POINT IS CHOSEN DEPENDS ON THE
C VALUE OF THE FUNCTION AND ITS GRADIENT AT X.
CALL FUNCTION(X,F,G)
27 IF (F-FX) 27, 27, 28
DA = C
CC291 = 1, A
CALFA = CALFA + G(I) * F(I)
IF(CALFA) 31, 33, 33
IF(F - FX) 32, 31, 33
IF(CX - CALFA) 32, 36, 32
FX = F
CX = CALFA
T = ALFA
AMBCA = ALFA
GC TC 22
IF(FY - F) 35, 34, 35
IF(CY - CALFA) 35, 36, 35
FY = F
CY = CALFA
AMBCA = AMBD - ALFA
GC TC 22
C COMPLETE DIFFERENCE VECTORS OF ARGUMENT AND GRADIENT FROM
C THE CONSECUTIVE ITERATIONS
CC37J = 1, N
K = N + J
P(K) = C(J) - H(K)
K = N + K
P(K) = X(J) - H(K)
C TERMINATE, IF FUNCTION HAS NOT DECREASED DURING LAST ITERATION
C IF(CLFF - F) 51, 51, 31
C TEST LENGTH OF ARGUMENT DIFFERENCE VECTOR AND DIRECTION VECTOR
C IF AT LEAST N ITERATIONS HAVE BEEN EXECUTED. TERMINATE, IF
C BOTH ARE LESS THAN EPS.
IER = 0
IF(KCLFN - N) 42, 35, 35
T = 0.
Z = 0.
C 4. J = 1, A
K = J + N
K = M(K)
K = K + N
T = T + A(F(S(K)))
Z = Z + H(K)
WRITE(6, 817) T, A
FORMAT(5F3.0, 1..5)
IF(T-A(N) < EPS) 41, 41, 42
IF(T-EPS) 56, 56, 42
C TERMINATE, IF NUMBER OF ITERATIONS WOULD EXCEED LIMIT
IF(KCUNT-LIMIT) 43, 5..50
C PREPARE UPDATING OF MATRIX H
ALFA = C
CC47J = 1, N
K = J + N
K = J + N
CC46L = 1, N
KL = N + L
L = W + H(KL) * H(K)
IF(L - J) 44, 45, 45
K = K + N - L
CC TO 46
K = K + 1
C CONTINUE
K = N + J
ALFA = ALFA + W * H(K)
H(J) = H
C REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF RESULTS
C ARE NOT SATISFACTORY
IF(Z - ALFA) 48, 1, 48
C UPDATE MATRIX H
K = N + 1
CC49L = 1, N
KL = N + L
CC45J=L,N
NJ=N2+J
H(K)=H(K)+P(KL)*F(NA)/Z-P(L)*F(J)/ALFA
49  K=K+1
CC TC 5
C     END OF ITERATION LOOP
C     AC CONVERGENCE AFTER LIMIT ITERATIONS
50   IER=1
RETURN
C     RESTORE CLC VALUES OF FUNCTION AND ARGUMENTS
51   CC52J=1,N
      LL = KR + J
52   X(J) = H(LL)
      F = CLCF
C     REPEAT SEARCH IN DIRECTION OF STEEPEST DESCENT IF DERIVATIVE
C     FAILS TO BE SUFFICIENTLY SMALL
      IF(CARM-EPS)55,55,52
C     TEST FOR REPEATED FAILURE OF ITERATION
53   IF(IER)56,54,54
54   IER=-1
      CC TC 1
55   IER=C
56   RETURN
END
MARQUARDT'S METHOD
C

MAPGUARDS METEOR FOR LEAST SQUARES OPTIMIZATION
C
C AS APPLIED TO THE SPECIFIC PROBLEM OF DETERMINING THE DYNAMIC
C
PARAMETERS.
C
COMMCA X(550), YA(550), N, KJK
DIMENSION G(6), A(6,6), IB(5), P(6), AA(6,6), DEL(6)
PLAN = 5.
ITR = 1
KJK = 0
CALL INPUT(K,IP,IP,P)

C XIA GENERATES EXPERIMENTAL DATA
CALL XIA(IN)
1)
CALL FTPXX(E,IP,K,IF,G,A,PHI)
CALL XNCKM(A,G,K,AA)
CALL LAMB(PLAN,AA,A,K,G,B,DEL,PHI)
WRITE(6,12) ITR, PHI, (B(I), I = 1, K), KJK, PLAN
1CC FORMAT(14P) ITERATION NO. 15/7 
PHI = E15.8/2H8.8,6F18.6
1/28HNSNO. CF FUNCTION EVALUATIONS 15/10L. LAMBDA = E15.8
WRITE(6,131) GAIN
131 FORMAT(5H: GAIN E15.8)
ITR = ITR + 1
CC 12 I = 1,K
12 IF(ABS(DEL(I)) .GT. .001) GO TO 1
CALL CLOCK(ICUT)
TIN = (ICUT - IA) / 133
WRITE(6,889) TIN
888 FORMAT(13H: RUNNING TIME E13.6, 2X,7HSECONDS)
STOP
END
SUBCTLIME INPUT(K,IP,IP,P)
COMMCA X(550), YA(550), N, KJK
DIMENSION IB(5), P(6)
C
K = TOTAL NUMBER OF PARAMETERS EVER USED

232
IP = NUMBER OF PARAMETERS USED IN ANY ONE PROBLEM
READ(5,10) K, IP
10 FORMAT(213)
C IE = IDENTIFICATION OF PARAMETERS NOT BEING USED
READ(5,11)(I0(I), I = 1,5)
11 FORMAT(513)
C B = INITIAL VALUES OF PARAMETERS BEING USED
READ(5,12)(B(I), I = 1, K)
12 FORMAT(6F10.5)
RETURN
END

SUBROUTINE LAMB(FLAM,A,A,K,C,B,DEL,PHI)
COMPLEX X(55.), Y(55.), N, K, J
DIMENSION A(6,6), A(6,6), G(6), P(6), PA(6), DEL(6), SQAA(6), GA(6)
V = 2.5
LW = 1
IF(FLAM/V .LT. 2.031) GO TO 6
ALAM = FLAM/V
NLX = 1
GO TO 130
6 ALAM = FLAM
NLX = 2
130 CC 1 I = 1, K
CC CC J = 1, K
10 A(A(I,J)) = A(I,J)
GA(I) = G(I)
AA(I,I) = A(I,I) + ALAM
1 SQAA(I) = SQRT(2*(I,1))
CALL GAUSS(K,DEL,AA,GA)
CC 2 I = 1, K
2 PA(I) = B(I) + DEL(I) / SQAA(I)
CALL SUBR2(PA,PHI)
WRITE(6,31) PHI

30 FORMAT(8H:PHI(I) = E20.8)
GC TC (7,8),NLX
7 IF(Phi(I) .LE. Phi(I) GC TC 20)
GC TC 6
8 IF(Phi(I) .LE. Phi(I) GC TC 20)
ALAM = ALAM * V*WLW
LW = LW + 1
GC TC 100
20 PLAM = ALAM
PHI = PHI
DC 21 I = 1,K
21 R(I) = RA(I)
RETURN
END

COMMON X(55:), Y(55:), N, KJK
COMMON CALCULATION OF FUNCTION, SUM SQ, CELTA F AND MATRIX
SUBROUTINE PTMPX(DP(6,B,P),P,G,A,PHI)
C START THE CALCULATION OF THE PTM MATRIX
DIMENSION G(6),F(6,B),IB(5),P(6),B(6),RA(6)
DIMENSION FP(7),SP(7),FX1(7),FX2(7)
COMMON X(55:),Y(55:),N,KJK,GAIN
940 FORMAT(13H Y(55:), N, KJK
WRITE(6,2) (R(I), I = 1,6)
2 FORMAT(6F20.5)
KJK = N + 4
G TC J = 1,7
FP(J) = 0.
SP(J) = 0.
FX1(J) = 0.
10 FX2(J) = 0.
DC 62 I=1,K
G (I) = 0
DC 62 J=1,K
IF (CEw .LT. .00001) DBw = .00001
THw = P(J)
B(J) = B(J) + DBw
INPER = 2
CC TC 2951
620 B(J) = THw
P(J) = (F-FWS)/CBw
CC TC 622
621 P(J) = C.
622 J = J + 1
IF (J-K) 628, 628, 624
624 F = FWS
END CF ESTIMATED PS ROUTINE
80 CC 82 JJ = 1, K
G(JJ) = G(JJ) + (Y(I) - F)*P(JJ)
CC 82 II = J, K
A(II, JJ) = A(II, JJ) + P(II)*P(JJ)
82 A(JJ, II) = A(JJ, II)
314 WS = Y(I+1) - F
IF (KJR .LT. 4) CC TC 315
PHI = PHI + WS*WS
315 I = I + 1
ILL = C
IF (I-N) 632, 632, 84
84 IF (IP) 88, 88, 85
85 CC 87 JJ = 1, IP
IWS = IE(JJ)
CC 86 II = J, K
A(1WS, II) = .
86 A(II, IWS) = .
87 A(IWS, IWS) = .
WRITE (6, 1) PHI
1 FORMAT(7F9.8)
88 RETURN
END

CCC   NORMA L IZ ES M AT RIX B EFORE I N V E R S I O N
SUBROUTINE XNCRF(A,E,K,AA)
DIMENSION A(6,6),E(6),AA(6,6),SA(6)
COMMON X(55), Y(55), N, KJN
90 CC  N2 I=1,K
IF(APS(A(I,I))-1.E-6) 5900,5900,5900
5900 A(I,I) = 0.
SA(I) = 0.
GO TO 5900
5901 SA(I) = SCRT(A(I,I))
92 CC  N5 I=1,K
GO TO 5901
96 A(I,I) = 0.
GO TO 92
98 A(I,J) = A(I,J)/KS
100 CC  N4 I=1,K
IF (SA(I)) 102,102,102
102 C(I) = C.
GO TO 104
104 C(I) = C(I)/SA(I)
106 CC  N6 I=1,K
110 A(I,I) = 1.
RETURN
END

SUBROUTINE SUBR2( P, PHI )
EVALUATION OF THE CRITERION FUNCTION FOR THREE DIMENSIONAL
SEARCH WITH THE SYSTEM GAIN BEING DETERMINED BY REGRESSION
ANALYSIS
COMMON X(550), YA(550), N, KJK
DIMENSION B(7), P(6)
NI = N + 1
KJK = KJK + 1
XY = 0.
SX = 0.
SXS = 0.
IF(KJK .NE. 5) GO TO 10
SY = 0.
SYS = 0.
DO 5 I = 2, NI
   SY = SY + Y(I)
5  SYS = SYS + Y(I)**2
   YS = SY**2
   YSC = SYS - YS/FLOAT(N)
10  DO 11 II = 1, 7
   11  B(II) = 0.
   DO 12 II = 1, 3
      L = II + 1
   12  B(L) = P(II)
   B(5) = 1.
940 FORMAT(13H YHAT TOC BIG)
300 I=1
   PHI = 0.
302 IF(I-1) 2900, 2900, 2901
2900 FX1 = 0.
   FX2 = 0.
   SFP = 0.
   FP = 0.
   FX1WS = 0.
   FPWS = 0.
FX2WS = 0.
SFPWS = 0.
2901 JB = B(2)
2001 FJB=JB
   IIJB=I-JB
   FLAG=(FJB+1.-B(2))*X(IIJB) +(B(2)-FJB)*X(IIJB-1)
   GO TO 2004
2002 FLAG=C.
   GO TO 2004
2003 FJB=JB
   IIJB=I-JB
   FLAG=(FJB+1.-B(2))*X(IIJB)
2004 FX3=B(3)*FX2-B(4)*FX1+B(5)*(1.-B(3)+B(4))*FLAG
2006 F=+FX3+Y(I )-(1.-B(6))*FP+B(7)*SFP
2007 FPWS = Y(I+1) - F
   IF(ABS (FPWS)-1.E+9 ) 2011,2011,5111
2011 SFPWS = SFPWS + FPWS
2012 IF(ABS (FX3)-1.E+9 ) 2013,2013,5111
5111 PHI = 1.E+18
WRITE *(6,940)
   I = N
   GO TO 2009
2013 CONTINUE
   FP = FPWS
   SFP = SFPWS
   FX1 = FX2
   FX2 = FX3
2009 CONTINUE
   SX = SX + F
   SXS = SXS + F*S
   XY = XY + F * Y(I+1)
I = I + 1
IF(I - N) 302,316,316

316 XS = SX*2
GAIN = (XY - SX * SY / FLOAT(N)) / (SXS - XS / FLOAT(N))
YHC = GAIN *(XY - SX * SY / FLOAT(N))
P(4) = GAIN
PHI = YSC - YHC
RETURN
END

SUBROUTINE GAUSS(N,G,AA,B)
COMMON X(550), YA(550), N, KJK
DIMENSION G(6), AA(6,6), B(6)
WRITE(6,79)
79 FORMAT(5HCAMAT)
DO 83 I = 1,N
83 WRITE(6, 80)(AA(I,J), J = 1,N)
80 FORMAT(IX,7E17.8)
WRITE(6, 81)(B(I), I = 1,N)
81 FORMAT(5H0BVEC/(IX, 7E17.8))
10 NN = N - 1
DO 5 I = 1,NN
J = N+1 - I
L = J - 1
DO 20 II = 1,L
IF(ABS(AA(J,II)) .GE. ABS(AA(II,J))) GO TO 20
DO 21 LL = 1,N
AS = AA(J,LL)
AA(J,LL) = AA(II,LL)
0 21 AA(II,LL) = AS
AS = B(II)
B(II) = B(J)
B(J) = AS
20 CONTINUE
**Subroutine XIN(NPAR)**

CENTER GENERATES DATA FROM THE DYNAMIC MODEL

COMMON X(550), YA(550), N, KJK
DIMENSION AB(7), XA(550), NPAR(6)
PY = 0.
READ(5,1) (AB(I), I = 1,7)
1 FORMAT(7F10.5)
XIMI = 0.
XI = 0.
YA(I) = 0.
XW = 0.
Q = 0.
N = AB(1)
NI = AB(1) + 1.5
DO 3 I = 1, NI
IF(I .EQ. 5) Q = 10.
XA(I) = Q
X(I) = XA(I) - XW
3 XW = XA(I)
   J = AB(2)
   FJ = J
   DO 100 I = 1, N
   K = I - J
   IF(K) 10, 10, 20
10 IF(K-1) 30, 30, 40
   20 FLAG = (FJ+1. - AB(2)) * X(K) + (AB(2) - FJ) * X(K-1)
   GO TO 60
10 FLAG = 0.
   GO TO 60
30 FLAG = (FJ + 1. - AB(2)) * X(K)
60 XIP1 = AB(3)*XI - AB(4)*XIM1 - AB(5)*(1. - AB(3)+AB(4)) * FLAG
   XIM1 = XI
   XI = XIP1
   YA(I) = -XIP1 + PY
   PY = YA(I)
100 WRITE(6, 2) I, XA(I), YA(I)
2 FORMAT(1HO, 15, 1G, 5H, XA = F20.4, 10X, 5HYA = F20.4)
RETURN
END
SUBROUTINE PATTERN(NP,P,STEP,NRC,IC,COST)
C
C
C PATTERN SEARCH PROGURARIOED BY C. F. MOORE   LSU
C
C NP = NUMBER OF PARAMETERS
C
C P = INITIAL VECTOR OF PARAMETERS. MUST BE WITHIN STABLE
C REGION
C
C STEP = INITIAL STEP SIZE
C
C NRC = NUMBER OF TIMES THE STEP SIZE IS REDUCED BE A
C FACTOR OF TEN
C
C IC = PRINT CONTROL
C
C IC = 0 NO PRINTING IS DONE
C
C IC = 1 ONLY THE ANSWER IS PRINTED
C
C IC = 2 PRINTING IS DONE FOR EACH ITERATION
C
C IC = 3 PRINTING IS DONE FOR EACH TRIAL STEP
C
C COST = VALUE OF THE CRITERION FUNCTION
C
C DIMENSION P(NP),STEP(NP),B1(100),B2(100),T(100),S(100)
C
C STARTING POINT
C
L=1
ICN=2
ITER=0
C5 I=1,NP
B1(I)=P(I)
B2(I)=P(I)
T(I)=P(I)
S(I)=STEP(I)*1.
C
C INITIAL BOUNDARY CHECK AND COST EVALUATION
C
C Bounds = User Supplied Subprogram Which Evaluates The Stability
C At Point P. If ICUT Goes Back 2 The Point Is In Bounds And
C If ICUT Is 1 The Point Is Out Of Bounds
C
CALL BOUNDSP(P,ICUT)
IF(ICOIN.E.0)GOTO12
IF(ICOIN.E.0)GOTO6
WRITE(6,1)S
12...
WRITE(6,131)(J,P(J),J=1,NI

10 CALL FRCC(P,C1)
   IF(IC.LE.0)GOTO11
   WRITE(6,131)ITIFR,C1
   WRITE(6,132)(J,P(J),J=1,NI

C------BEGINNING OF PATTERN SEARCH STRATEGY
11 CC99 INRD=1,ARC
   CC12 I=1,NI
12 S(I)=S(I)/10.
   IF(IC.LE.0)GOTO20
   WRITE(6,132)
   WRITE(6,133)(J,S(J),J=1,NI

20 IFAIL=4.0

C------PRETURBATION ABOUT T
   CC30 I=1,NI
   IC=O
21 P(I)=T(I)+S(I)
   IC=IC+1
   CALL ECUNDS(P,ICLT)
   IF(ICLT.GT.1)GCTC23
   CALL FRCC(P,C2)
      L=L+1
      IF(IC.LT.3)GOTO22
      WRITE(6,132)L,C2
      WRITE(6,135)(J,P(J),J=1,NI

22 IF(C1-C2)23,23,25
23 IF(IC.GE.2)GCTC24
   S(I)=-S(I)
   GCTC21
24 IFAIL=IFAIL+1
   P(I)=T(I)
   GCTC32
25 T(I)=P(I)
C1=C2

30 CONTINUE
IF (IFAIL.LT.NP) GOTO 35
IF (ICK.EQ.2) GOTO 39
IF (ICK.EQ.1) GOTO 35
CALL PRC(T,C2)

L=L+1
IF (IC.LT.2) GOTO 31
WRITE(*,2) L,C2
WRITE(*,2) (J,T(J),J=1,NP)

31 IF (C1-C2)32,34,34
32 ICK=1
CC33 I=1,NP
B1(I)=B2(I)
P(I)=P2(I)
33 T(I)=E2(I)
CC3C
34 C1=C2
35 IP1=C
CC39 1=1,NP
P2(I)=T(I)
IF (ABS(B1(I)-B2(I)).LT.1.0E-2) IB1=IB1+1
39 CONTINUE
IF (IE1.EQ.NP) GOTO 39
ICK=C

ITER=ITER+1
IF (IC.LT.2) GOTO 42
WRITE(*,1) ITER,C1
WRITE(*,1) (J,T(J),J=1,NP)

C----- ACCELERATION STEP
40 SJ=1.0
CC45 1J=1,11
CC42 I=1,NP
T(I)=E2(I)+SJ*(F2(I)-F1(I))
POWELL'S METHOD
SUBROUTINE PCW(FP, NF, 38, AM, EP, NSTEP, NPRINT)
C  PCW ELLS METHOD OF OPTIMIZATION
C  FP = VECTOR OF INITIAL PARAMETERS
C  NF = NUMBER OF PARAMETERS
C  NSTEP = NUMBER OF TIMES STEP SIZE IS REDUCED BY A FACTOR OF TEN
C  AM = MAXIMUM STEP SIZE ALLOWED
C  EF = INITIAL INTERVAL OF CONVERGENCE. WHEN STEP SIZE IS
C  CHANGED THE INTERVAL OF CONVERGENCE IS REDUCED BY A FACTOR
C  OF TEN
C  NPRINT = 0 ONLY THE FINAL ANSWER IS PRINTED
C  NPRINT = 1 INFORMATION IS PRINTED AT EACH ITERATION
C  EF = .1 * AM
C  KJK = ITERATION NUMBER
DIMENSION P(4,11), S(1:11), PP(1:1), B(3)
C  MUST CHANGE DIMENSION STATEMENTS FOR MORE THAN 10 VARIABLES
NF = 1
NFP = NF + 1
ITR = 0
CC 1C J = 1,4
CC 1C I = 1, NPP
10 P(J,I) = 0
CC 8 J = 1, NF
C  P MATRIX WHICH CONTAINS FOUR PARAMETER VECTORS. THE BEST
C  VECTOR IS ALWAYS FOUND IN ROW FOUR.
C  S MATRIX OF DIRECTIONS. INITIALLY THE COORDINATE DIRECTIONS
CC 9 I = 1,NPP
9 S(J, I) = 0
S(J, J) = 1.
S(J, NFP) = 1.
8 P(4,J) = PP(J)
CALL FRCC(P, 4, NF)
PP(NF+1) = P(4, NF+1)
FCLC = P(4, NF+1)

249
18 DELTA = 0.
   ITR = ITR + 1
   CC 17 J = 1, NP
17 PP(J) = P(4, J)
   F1 = FP(NP)
   CC 20 J = 1, NP
   K = J
   CALL SRCH(S, P, K, NP, PR, AM, FP, P)
   TDELTA = POLC - P(4, NP+1)
   MT = J
   IF(TDELTA .LT. DELTA) GO TO 15
   DELTA = TDELTA
   N = MT
19 POLC = P(4, NP+1)
20 CONTINUE
   CC 25 J = 1, NP
25 P(3, J) = 2. * P(4, J) - PP(J)
   CALL FRCC(P, 3, NP)
   F3 = F(3, NP+1)
   F2 = F(4, NP+1)
   IF(F2 .GE. F1) GO TO 48
   IF((F1 - 2.*F2+F3)*((F1-F2-DELTA)*F2) .GE. (0.5*DELTA)*(F1-F3)**2)
1 CC TC 48
   IF(M .GE. NP) GO TO 29
   NPR = NP - 1
   CC 30 J = M, NPR
   CC 32 I = 1, NP
30 S(J, I) = S(J+1, I)
29 SUM = 0.
   CC 31 J = 1, NP
   S(NP, J) = P(4, J) - PP(J)
31 SUM = SUM + S(NP, J)**2
   SUM = SCRT(SUM)
   CC 32 J = 1, NP
32 $S(NP, J) = S(NP, J) / SUP$
$S(NP, NP + 1) = 3.3$
CC 32 $J = 1, NP$
$P(2, J) = P(4, J)$
$P(4, J) = PP(J)$
33 $P(1, J) = PP(J)$

C ACCELERATION STEP

B(1) = 0.
B(2) = $(P(2, 1) - P(1, 1)) / S(NP, 1)$
B(3) = $Z * B(2)$
CALL SRCH(S, P, NP, NP, PP, AM, EP, B)
$S(NP, NP + 1) = 1.$
PCLC = $P(4, NP)$

48 IF(NPNT .EQ. 2) GO TO 49
WRITE(6, 74) ITR
WRITE(6, 40) (P(4, J), J = 1, NP)
WRITE(6, 41) P(4, NP + 1)
DC 111 I = 1, 4
111 WRITE(6, 112) (P(I, J), J = 1, NP)
112 FORMAT(2I10, 7F16.5)
CC 115 I = 1, NP
115 WRITE(6, 113) (S(I, J), J = 1, NP)
113 FORMAT(2I10, 7F16.5)
WRITE(6, 73) KJK
49 CC 5C. J = 1, NP
50 IF(APS(P(4, J) - FP(J)) .GT. EP) GO TO 18
IF(NAST .EQ. NST(P)) CC TC 54
BR = EB / 2.
AF = AM / 2.
NAST = NAST + 1
CC 1C 18
54 WRITE(6, 73) P(4, AF + 1)
WRITE(6, 74) ITR
SUBROUTINE SRCF(S,P,NC,NP,RR,AM,EP,B)
C CAN DIMENSIONAL SEARCH
C USING THREE POINTS A QUADRATIC EQUATION IS FITTED,
C THE TURNING POINT IS PREDICTED AND IF IT DOES NOT EXCEED THE
C MAXIMUM STEP ALLOWED IT REPLACES THE WORST POINT. THIS IS
C REPEATED UNTIL TWO POINTS LIE WITHIN THE FINAL INTERVAL OF
C UNCERTAINTY
DIMENSION L(3),P(2,11),S(13,11)
COMMON X(5),Y(5),N,KJ,KAIN
RST = 0
SAM = AM
CC 7 J = 1,3
7 L(J) = 0
  IF(S(NP,NP+1) .NE. 2.) GO TO 5
    L(1) = 1
    L(2) = 2
    L(3) = 3
    SAM = 3. * P(2)
    CC TO 100
5 P(1) = .
CC 8 J = 1, NP
8 P(1,J) = P(4,J)
P(1, AF+1) = P(4, AF+1)
L(1) = 1
B(2) = EB
CC J = 1, NP

10 P(2, J) = P(4, J) + E(2) * S(NC, J)
CALL FRCC(P, 2, AF)
L(2) = 2
IF(P(1, AP+1) .LE. P(2, NP+1)) CC TC 21
B(3) = 2 * EE
CC J = 1, NP

20 P(3, J) = P(4, J) + B(3) * S(NC, J)
CALL FRCC(P, 3, AF)
L(3) = 3
CC TC 10

21 B(3) = -BR
CC J = 1, NP

22 P(3, J) = P(4, J) + B(3) * S(NC, J)
CALL FRCC(P, 3, AF)
CC J = 1, 3

23 L(J) = L(J) + 1
100 KST = KST + 1

IF(KST .GE. 11) CC TO 7

C = 5 * ((B(2) * E(3) * E2) * P(1, AP+1) + (B(3) * E2 - B(1) * E2) * P(2, NP+1) +
1 * (E(1) * E2 - B(2) * E2) * F(F, NP+1))
CD = (B(2) - B(3)) * P(1, NP+1) + (E(3) - E(1)) * P(2, NP+1) + (B(1) - B(2)) *
1 * P(3, AF+1)

E = C / CC
C2 = CD / (P(1) - B(2)) * (B(2) - P(3)) * (E(2) - E(1))
IF(C2 .GE. 1.) CC CC / ABS(B)

IF(APS(C) .GE. SAK + 1.) CC TC 3.

CC = C * SAK / ABS(C)
SAK = SAK + 1
CC TC 4.

30 CC J = 1, 3
31 IF (AES (P(J) - E) .LT. EP) GO TO 70
40 CC 41 J = 1, 3
   IF (L(J) .EQ. 1) NA = J
   IF (L(J) .EQ. 2) NP = J
41 IF (L(J) .EQ. 3) AC = J
   IF (B(NP) .LT. C) CC TO 50
   IF (B(NA) .GT. C) CC TO 45
   B(NC) = C
   CC 42 J = 1, NP
43 P(NC, J) = P(4, J) + P(NC) * S(NC, J)
   CALL FRCC(P, NC, NP)
   L(NA) = 1
   L(NC) = 2
   L(NB) = 3
   CC TC 1,
45 IF (P(NA, NP+1) .GT. P(NC, NP)) CC TO 48
   B(NC) = C
   CC 46 J = 1, NP
46 P(NC, J) = P(4, J) + P(NC) * S(NC, J)
   CALL FRCC(P, NC, NP)
   L(NC) = 1
   L(NA) = 2
   L(NB) = 3
   CC TC 1,
48 P(NA) = C
   CC 49 J = 1, NP
49 P(NA, J) = P(4, J) + P(NA) * S(NC, J)
   CALL FRCC(P, NA, NP)
   CC TC 1,
50 IF (P(NC) .LT. E) CC TO 55
   B(NA) = C
   CC 51 J = 1, NP
51 P(NA, J) = P(4, J) + P(NA) * S(NC, J)
   CALL FRCC(P, NA, NP)
L(NA) = 1
L(NC) = 2
L(NA) = 3
G(TC) = L(NF)

55 IF(P(NC, NP+1) .GT. P(NA, NP+1)) GC TO 58
   E(NA) = C
   GC 56 J = 1, NP

56 P(NA, J) = P(4, J) + P(NA) * S(NC, J)
   CALL FRC(P, NA, NP)
   L(NF) = 1
   L(NC) = 2
   L(NA) = 3
   GC TO 10

58 E(NC) = C
   GC 59 J = 1, NP

59 P(NC, J) = P(4, J) + E(NC) * S(NC, J)
   CALL FRC(P, NC, NP)
   GC TO 10

70 PT = F(1, NP+1)
   NL = 1
   CC 71 J = 2, 3
   IF(PT .LT. P(J, NP+1)) GC TO 71
   PT = F(J, NP+1)
   NL = J

71 CONTINUE
   GC 72 J = 1, NP

72 P(4, J) = P(NL, J)
   P(4, NP+1) = P(NL, NP+1)
RETURN
END
POWELL'S METHOD

WITH FIBONACCI SEARCH
SUBROUTINE PCOF(PP, NP, DEL, FIN, NC, NSTEP, NPRNT)
C POWELL'S METHOD OF OPTIMIZATION
C OPEN ENDED FIBONACCI SEARCH USED INSTEAD OF ONE SUGGESTED
C BY POWELL
C NP = NUMBER OF PARAMETERS
C NSTEP = NUMBER OF TIMES THE STEP SIZE IS REDUCED BY A FACTOR
C OF TEN
C NC = MAXIMUM NUMBER OF EXPERIMENTS PER ITERATION
C DEL = FINAL INTERVAL OF UNCERTAINTY
C FIN = INITIAL STEP SIZE
C NPRNT = 0 ONLY ANSWER IS PRINTED
C NPRNT = 1 INFORMATION AT EACH ITERATION IS PRINTED
C PP = VECTOR OF INITIAL PARAMETERS
DIMENSION P(5,11), S(10,11), PP(10), B(4), A(100)
A(1) = 1.
A(2) = 1.
DO 3 I = 3, 100
A(I) = 0.
3  A(I) = A(I-1) + A(I-2)
NST = 1
NPP = NP + 1
ITR = 0
DO 10 J = 1, 5
10  P(J,I) = 0.
C PMATRIX CONTAINS FIVE VECTORS OF PARAMETERS
C THE BEST PARAMETER VECTOR IS ALWAYS IN THE FIFTH ROW
C SMATRIX CONTAINS THE SEARCH DIRECTIONS
C INITIALLY THE COORDINATE DIRECTIONS ARE USED
DO 8 J = 1, NP
DO 9 I = 1, NPP
8  S(J,I) = 0.
9  S(J,J) = 1.
S(J,NP) = 1.
8  P(5,J) = PP(J)
    CALL FRCC(P,5,NP)
    PP(NP+1) = P(5,NP+1)
    FCLE = P(5,NP+1)
18  DELTA = 5.

C
ITR = ITERATION NUMBER
ITR = ITR + 1
CC 17  J = 1,NP
17  PP(J) = P(5,J)
    F1 = PP(NP)
    CC 2C  J = 1,NP
    K = J
    CALL SRCH(5,P,K,NP,CEL,FIN,NC,A)
    TCELT A = BCLE - P(5,NP+1)
    MT = J
    IF(TCELT A .LT. DELTA) GC TG 15
    DELTA = TCELT A
    M = MT
19  BCLCE = P(5,NP+1)
20  CONTINUE
CC 25  J = 1,NP
25  P(3,J) = 2.* P(5,J) - PP(J)
    CALL FRCC(P,3,NP)
    F3 = P(3,NP+1)
    F2 = P(5,NP+1)
    IF(F3 .GE. F1) GC TC 48
    IF((F1-2.*F2+F3)*((F1-F2-DELTA)*=2) .GE. (.5*DELTA)*F1-F3)*=2)
1CC IF 48
    IF(M .GE. NP) GC TC 29
    APR = NP - 1
    CC 30  J = APR,NP
    CC 3C  I = 1,APR
30  S(J,I) = S(J+1,1)
29 SUM = .
   CC 31 J = 1, NP
   S(NP,J) = P(5,J) - FP(J)
31 SUM = SUM + S(NP,J)*P
   SUM = SCRT(SUM)
   CC 32 J = 1, NP
32 S(NP,J) = S(NP,J) / SUM
   S(NP,NP) = 3.0
   IF(P(E,APP) .GT. P(E,APP)) CC TO 37
   CC 35 J = 1, NP
35 P(5,J) = P(3,J)
37 CALL SRCH(S,P,NP,APP,CPL,FIN,AC,A)
   S(NP,AP+1) = 1.0
   PCRLD = P(5,APP)
48 IF(NAFFNT .EQ. C) CC TO 49
   WRITE(6,74) ITR
   WRITE(6,47) (P(E,J), J = 1, NP)
   WRITE(6,41) P(5,AP+1)
   CC 111 I = 1, 5
111 WRITE(6,112) (P(1,J), J = 1, APP)
112 FORMAT(2H,P,7F16.5)
   CC 115 I = 1, NP
115 WRITE(6,113) (S(1,J), J = 1, APP)
113 FORMAT(2H,S,7F16.5)
   WRITE(6,73) KJK
49 CC 50 J = 1, NP
50 IF(AABS(P(4,J) - FP(J)) .GT. FIN ) GO TO 18
   IF(NST .NE. NSTFP) CC TO 54
   FIN = FIN / 1.0
   NST = NST + 1
   CC TO 18
54 WRITE(6,74) P(5,AP+1)
   WRITE(6,74) ITR
   CC 71 J = 1, NP
This subprogram performs a one-dimensional search. Steps are taken in the search direction in an accelerated manner until a minimum is bracketed. A Fibonacci search is then used to locate the minimum.

Routine SRCH(S, P, ND, NP, DEL, FIN, N, A)

DIMENSION B(4), P(5,11), S(10,11), A(100)
NPP = NP + 1
B(1) = 0.
DO 8 J = 1, NPP
  P(1,J) = P(5,J)
  B(3) = (A(N-2) - A(N-4) * DEL) * FIN
  DO 11 J = 1, NP
  11 P(3,J) = P(5,J) + B(3) * S(ND,J)
  CALL PROC(P,3,NP)
  IF (ABS(P(1,NPP) - P(3,NPP)) .GT. DEL) GO TO 15
  B(4) = B(3)
  DO 12 J = 1, NPP
  12 P(4,J) = P(3,J)
  B(3) = (A(N-3) - A(N-5) * DEL) * FIN
  DO 13 J = 1, NP
  13 P(3,J) = P(5,J) + B(3) * S(NC,J)
  CALL PROC(P,3,NP)
NST = N - 5
15 IF(P(3,APP) .LT. P(1,APP)) GC TC 25
   GC 16 J = 1,APP
   PT = F(1,J)
   P(1,J) = P(3,J)
   P(3,J) = PT
16 P(5,J) = P(1,J)
   B(3) = -B(3)
   SS = -1.
   GC IC TC 3:
25 SS = 1.
30 B(4) = SS * (A(N-1) - A(N-3) * DEL) * FIN
   GC 31 J = 1,AP
31 P(4,J) = P(5,J) + P(4) * S(NC,J)
   CALL FRCC(P,4,AF)
   IF(P(4,APP) .LT. P(2,APP)) GC TC 49
   NST = N - 4
   GC TC 1?
46 L = 2
80 L = L + 1
   B(1) = B(3)
   B(3) = B(4)
   GC 81 J = 1,APP
   P(1,J) = P(2,J)
81 P(3,J) = P(4,J)
   NPL = N + L
   B(4) = SS * (A(NPL-4) - A(NPL-6) * DEL) * FIN + B(1)
   GC 82 J = 1,AP
82 P(4,J) = P(5,J) + E(4) * S(NC,J)
   CALL FRCC(P,4,AF)
   IF(P(4,APP) .LT. P(2,APP)) GC TC 39
   NST = N + L - 7
   B(3) = SS * (A(NST+2) - A(NST) * DEL) * FIN + B(1)
   GC 83 J = 1,AP
83 P(3, J) = P(5, J) + P(2) - S(N, J)
    CALL FPCC(P, 3, NF)
100 P(2) = P(1) + E(4) - A(2)
    CALL FPCC(P, 2, AF)
101 P(2, J) = P(5, J) + P(2) * S(N, J)
    CALL FPCC(P, 2, AF)
    NCBS = 1
115 IF(NCES .EQ. NSE) GC TC 190
    IF(P(2, APP) .LT. P(3, APP)) GC TC 150:
    B(1) = E(2)
    B(2) = E(3)
    CALL J = 1, NP
    P(1, J) = P(2, J)
120 P(2, J) = P(3, J)
    B(3) = P(1) + E(4) - A(2)
    CALL J = 1, NP
121 P(3, J) = P(5, J) + P(2) * S(N, J)
    CALL FPCC(P, 3, NF)
    NCBS = NCBS + 1
    GC TC 115
150 B(4) = P(3)
    B(3) = E(2)
    CALL J = 1, AFF
    P(4, J) = P(3, J)
151 P(3, J) = P(2, J)
    B(2) = C(1) + E(4) - A(2)
    CALL J = 1, NP
152 P(2, J) = P(5, J) + P(2) * S(N, J)
    CALL FPCC(P, 2, AF)
    NCBS = NCBS + 1
    GC TC 115
190 IF(P(2, APP) .LT. P(3, APP)) GC TC 195
    CALL J = 1, AFF
191 P(5, J) = P(3, J)
RETURN
195 CC 156 J = 1,MPF
196 P(5,J) = P(2,J)
   RETURN
   END
ROSENROCK'S METHOD I
SUBROUTINE ROSENBROCKS(FILL CLIMBING METHOD 1)
C
X = PARAMETERS OF PROBLEM
C
U = MATRIX USED BY ROSENBROCKS METHOD
C
H = .1
C
VV = VALUE OF THE CRITERION FUNCTION
C
NFL = ITERATION NUMBER
C
N = NUMBER OF PARAMETERS
C
VF = USER SUPPLIED FUNCTION PROGRAM TO EVALUATE THE
C
CRITERION FUNCTION
C
AT THE END OF EACH ITERATION THIS PROGRAM MUST BE CALLED AGAIN.
C
CONVERGENCE DID NOT OCCUR THIS SUBROUTINE MUST BE CALLED AGAIN

DIMENSION U(1:N), X(1:N), A(1:N), R(1:N), UA(15,17), NS(26)

AA=3.
B=1.

IF(NFL=1)15,15,17
15 CC 24 I=1,N
C (I)=.
CC 25 J=1,N
25 U(I,J)=.
24 U(I,1)=1
VV = VF
17 CC 1 J=1,N
NS(J)=.
L=1
E=H
VV = VV
VVT=VV
2 CC 3 I=1,N
3 E=U(1)+C[U(J,1)]
CV = VF - VV
VV = VV + CV
CC TC (4,5,76,27),L
Z = 0
CC 12 K = 1, N
V = 0
CC 13 L = M, N
13 V = V + C(L) * U(1, K)
12 Z = Z + L * N(J, K) * V
A(V, J) = Z
11 CONTINUE
DC 14 J = 1, M
14 Z2 = Z2 - A(N, J) * UM(J, I)
32 R(1) = Z2
6 C = C + Z2 * s * 2
C = 1. / SCRT(C)
CC 1C I = 1, N
19 UM(K, I) = C * R(1)
9 CONTINUE
DC 18 I = 1, N
U(I) = C.
DC 15 J = 1, N
19 U(I, J) = LN(1, J)
18 CONTINUE
RETURN
END
ROSENBRICK'S METHOD II
SUBROUTINE PSCE(P,PHI)
C---CRITIMIZATION PROGRAM
C---PSCEBICK BELL CLIFFING METHOD II
C
P = PARAMETERS OF THE PROBLEM
C
PHI = VALUE OF THE CRITERION FUNCTION
DIMENSION X(IC),V(IN),E(IN),K(IN),XO(IN),B(IN),P(IN),
IA(IN),AL(IN),XC(IN),FC(IN)
C---INFORMATION REQUIRED IN THE PROGRAM
C
IMAX = MAXIMUM NUMBER OF ITERATIONS
IMAX=100
EXIT=0
C
FIN = CONVERGENCE CRITERION
FIN=0.001
C
KP MUST BE SET EQUAL TO THE NUMBER OF PARAMETERS
KP=AFMT
CC51=1,KP
XC(I)=P(I)
5
X(I)=P(I)
C---CALCULATE INITIAL VALUE OF CRITERION FUNCTION
C
PRCC = USER SUPPLIED SUBROUTINE TO EVALUATE CRITERION FUNCTION
CALL PRCC(X,FC)
CC70K=1,KP
CC70L=1,KP
70 V(I,K)=0.
IS=0
IU=0
IC=0
CC71J=1,KP
71 V(J,J)=1.
C---INITIALIZE FOR CURRENT COORDINATE SYSTEM
87 IC=0
CC72I=1,KP
E(I)=E.1
C(I)=C.1
72  N(I)=N
C---START CALCULATIONS
59  K=1
66  CONTINUE
67  G731=1,KP
73  X(R)=X(I)+F(K)*V(I,K)
C CHECK STABILITY LIMITS
C INSERT STABILITY LIMITS HERE
C---CALCULATE CRITERION FUNCTION
CALL PRC(F1,F)
GC TC 26
25  F = 1.E+33
26  IF(F-FC)52,53,51
51  E(K)=-E(K)/2.
52  IF(M(I)-3) 52,52,52
54  N(K)=C
55  IC=IC+1
GCTE53
50  G741=1,KP
74  X(IC)=X(I)
IS=IS+1
G(K)=G(K)+E(K)
E(K)=2.*E(K)
FC=F
IF(M(K)-3) 53,55,55
55  N(K)=2
GCTE53
53  IF(K-KP) 52,57,57
56  IF(K-KP) 53,55,55
58  K=K+1
CCTC6C
C---ROTATE COORDINATE SYSTEM AND CONSTRUCT ALPHA VECTORS
57 CONTINUE
C---END PROGRAM BY SPECIFYING THE MAX NUMBER OF ITERATIONS
   IF(IC.GE.ICMAX) CCTC94
C---END PROGRAM IF PARAMETERS ARE NOT BEING CHANGED APPRECIABLY
   CC7J=1,KP
7   CEL(J)=ABS((X(J)-XC(J))/X(J))
   CC8J=1,KP
   IF(CEL(J).LT.FINT) NEXIT=NEXIT+1
8 CONTINUE
   IF(NEXIT.EQ.KP) CCTC23
   CC9J=1,KP
9   XC(J)=X(J)
   CC75J=1,KP
   CC75I=1,KP
   S = C.
   CC 76 K = J,KP
76 S = S + C(K)*V(I,K)
75 A(I,J) = S
C
C NEARLY IDENTICAL
CC 77 J = 1,KP
   DUM = C.
   CC 78 K = 1,KP
78 DUM = DUM + A(K,J)*A(K,J)
   CC 79 I = 1,KP
79 AL(I,J)=AL(I,J)/SCR1(DUM)
77 CONTINUE
C---CONSTRUCT BETA VECTORS
   CC8.K=1,KP
   KFR=K-1
   CC9I1=1,KP
81 R(I)=AL(I,K)
   IF(KFR) C2,61,F2
APPENDIX D

COMPUTER PROGRAM FOR PATTERN SEARCH ON SIX PARAMETER MODEL
PATTERN SEARCH

DETERMINATION OF 6 PARAMETER STOCHASTIC PLUS DYNAMIC MODEL

COMMON A15, 80, Y(15), NPAR(6), N, NPAR, npar, NPARM, NPAR, Y(15)

DIMENSION PAR(6), STEP(6), XL(6), U(6)

DIMENSION YUF(51,51), XI(51,51), I11(5), I11(2)

CALL FPTARP(-3)
K = 200
KRP = 0

200 CALL CLOCK(IN)
K = K

NPARAT = NUMBER OF PARAMETERS

NSTEP = NUMBER OF TIMES THE STEP SIZE IS REDUCED

REAC(E, E) NPARAT, NSTEP

80 FORMAT(211,1)
IF(NPARAT .EQ. 3) GO TO 170

NPAR = PARAMETER IDENTIFICATION NUMBERS

REAC(E, 85) (NPAR(1), I = 1, NPARAT)

85 FORMAT(713)

PAR = INITIAL VALUES OF THE PARAMETERS. MUST BE WITHIN

STABLE REGION

REAC(E, 82) (PAR(1), I = 1, NPARAT)

82 FORMAT(7F11.5)

STEP SIZE

CC 81 I = 1, NPARAT

81 STEP(1) = .55

1002 CALL MODEL

CC 321 I = 1, 221

321 WRITE(6, 221) A(11), Y(11)

321 FORMAT(2F25.5)

CALL FVAR(E(NPAR), PAR, STEP, NPARAM, I, PHI)

KRP = 10

CALL FREG(PAR, PHI)

KRP = 1

274
SUBROUTINE PATTERN(IFP,P,STEP,ARC,IC,COST)
C     PATTERN SEARCH PROGRAMMED BY C. F. MOORE     LSU
C     IFP = NUMBER OF PARAMETERS
C     P = INITIAL VECTOR OF PARAMETERS. MUST BE WITHIN STABLE
C     REGION
C     STEP = INITIAL STEP SIZE
C     ARC = NUMBER OF TIMES THE STEP SIZE IS REDUCED BY A
C     FACTOR OF 1/2
C     IC = PRINT CONTROL
C     IC = 0 NO PRINTING IS DONE
C     IC = 1 ONLY THE ANSWER IS PRINTED
C     IC = 2 PRINTING IS DONE FOR EACH ITERATION
C     IC = 3 PRINTING IS DONE FOR EACH TRIAL STEP
C     COST = VALUE OF THE CRITERIA FUNCTION.
DIMENSION IFP,P(1P),STEP(1P),ARC(1P),IC(1P),C(1000)
C--- STARTING POINT
I=1
ICK=2
ITER=1
CC5 I=1,NP
B1(I)=P(I)
B2(I)=P(I)
T(I)=F(I)
S(I)=STEP(I)*1.
C-----INITIAL BOUNDARY CHECK AND COST EVALUATION
CALL FCKGS(P,1,CUT)
IF(1€5.U.E..)GO TO 11
IF(1€5.U.E..)GO TO 11
WRITE(6,10,5)
WRITE(6,10,6)(J,P(J),J=1,NP)
10 RETURN
11 CALL F2CG(P,C1)
IF(1€5.U.E..)GO TO 12
WRITE(6,10,7)(ITER,C1)
WRITE(6,10,8)(J,P(J),J=1,NP)
12 RETURN
C------BEGINNING OF PATTERN SEARCH STRATEGY
13 CCP3 I=1,ARC
CC12 I=1,NP
14 S(I)=S(I)/1.8
IF(1€5.U.E..)GO TO 15
WRITE(6,10,9)
WRITE(6,10,10)(J,S(J),J=1,NP)
15 RETURN
C------PRECONDITION ABOUT I
CC30 I=1,NP
IC=-;
20 IFAIL=1.8
C------PRECONDITION ABOUT I
CC30 I=1,NP
IC=-;
21 P(I)=1(I)+S(I)
!IC=IC+1
CALL FCKGS(P,1,CUT)
IF(I1=I2), GCTC23
CALL FRCC(P,C2)
    L=L+1
    IF (IC, L, 3) GCTC22
    WRITE(6,122)(L,C2)
    WRITE(6,123)(J,P(J),J=1,NP)
22 IF(C1-C2)23,23,25
23 IF(IC,GE,2)GCTC24
S(1)=-S(1)
GCTC21
24 IFAIL=IFAIL+1
P(1)=I(1)
GCTC3C
25 T(1)=P(1)
C1=C2
30 CONTINUE
IF(IF/1L,LT,NP)GCTC25
IF(ICK,EC,2)GCTC5
IF(ICK,EQ,1)GCTC35
CALL FRCC(T,C2)
    L=L+1
    IF(IC, L, 2) GCTC31
    WRITE(6,122)(L,C2)
    WRITE(6,123)(J,T(J),J=1,NP)
31 IF(C1-C2)32,34,34
32 ICK=1
CC33 I=1, NP
B1(I)=B2(I)
P(1)=C2(I)
33 T(1)=C2(I)
GCTC2:
34 C1=C2
35 IPI=
CC35 I=1, NP
**C-----ACCELERATION STEP**

40  SJ=1.
41  CC46  IT=1,11
42  CC42  I=1,NP
43  T(I)=F2(I)+SJ*(F2(I)-A1(I))
44  P(I)=T(I)
45  SJ=SJ-.1
46  CALL ECUNEO(T,ICT1)
47  IF(ICT1.LT.1)CC4C4E
48  IF(I1.EQ.11)ICK=1
49  CONTINUE
50  CC47  I=1,NP
51  P1(I)=P2(I)
52  CCCTC2C
53  CC51  I=1,NP
54  T(I)=F2(I)
55  CONTINUE
56  CC180  I=1,NP
57  P(I)=T(I)
58  CCST=Cl
59  IF(I1.LT.1)RETURN
60  WRITE(1,1441),Cl
61  WRITE(1,1441)(J,P(J),J=1,NP)
RETURN
6200  FORMAT(1X,5(17,E13.6))
SUBROUTINE BOUNDS(X,Y)
C BOUNDS CHECKS THE STABILITY LIMITS OF 6 PARAMETER MODEL
DIMENSION X(6)
I = 0
IF(X(1) .LT. -1.) GO TO 170
IF(X(2) .GT. X(2) + 1.) GO TO 170
IF(X(2) .LT. -(2*2.1)) GO TO 170
IF(X(3) .GT. 1.) GO TO 170
IF(X(4) .LT. 1.) GO TO 170
IF(X(5) .LT. 1.) GO TO 170
IF(X(6) .LT. 1.) GO TO 170
IF(X(5) .GT. (4.-X(6))/2.) GO TO 170
RETURN
170 I = 1
RETURN
END

SUBROUTINE PREC(F,P1)
C PREC EVALUATES THE CRITERION FUNCTION AT POINT B
C THE VALUE OF THE CRITERION FUNCTION P1 IS RETURNED
COMMON X(5), Y(6), SQA(A), N, KJK, NP, KKP, YA(5,7)
DIMENSION B(7), F(6)
YA(1) = 1.
KJK = KJK + 1
GO 11 11 = 1,7
11 F(11) = 1.
2011 SFRS = SFPS + FPWS
2012 IF(ABS(FX3)-1.E-9) 2013,2013,5111
5111 PHI = 1.4+16
          WRITE
          I = N
          GC TO 2:69
2013 CONTINUE
          FP = FPWS
          SFP = SFPS
          FX1 = FX2
          FX2 = FX3
2019 CONTINUE
304 PHI = PHI + (Y(141) - F)*12
          I = I+1
          IF(KRF .NE. 16) GC TO 302
          YA(I) = F
303 IF(I - N) 312,312,316
316 CONTINUE
          RETURN
          ENDD!
SUBROUTINE MODEL
DIMENSION R0H(4,2,1), R1(7,5)
CUMOCA X(5,2), Y(5,1), M(4,6), R0H(4,2,1), R1(7,5)
C C GENERATE NOISE FOR REACTOR FEED STREAMS
C RFS = ROOT MEAN SQUARE VALUE OF NOISE
CC 56 J = 1,4
C XC AND YC NUMBERS TO INITIALIZE RANDOM NUMBER GENERATOR
C RAN = 1.
C T = TIME CONSTANT OF FIRST ORDER LAG USED TO GENERATE NOISE
C RFS = DESIRED ROOT MEAN SQUARE VALUE OF NOISE
C PC = STEP INPUT FLIGHT
C K = NUMBER OF RANDOM NUMBERS TO GENERATE
7)

```fortran
REAL(5,4) XC,YC,ANF,E,ANS,PC,AN
4 FORMAT(6F15.5,1x)
ZP = RAND(XC,YC)
XC = YC
AI = A * 10
NIP = AI + 1
CC S I = 1,NIP
R(I) = RAND(XC,YC) * ANF + 2. - ANF
3 IF(T.FC,J.) PITA(J,J) = R(I)
   KKK = 1
   G = 1.
53 NT = 5
   E1 = 6.
   CA = 6.
   DT = .2
   IF(T.FC,J.) CC TC 57
   CC 5S I = 1,NT
   BETA(J,J) = CA
   FLAG = R(I)
   CR = (R(I+1) - R(I)) / FLOAT(NT)
   CC 5S LL = 1,NT
   CA = CA + (C*FLAG - CA) / T * CT
51 FLAG = FLAG + CR
50 E1 = E1 + BETA(J,J)*2
59 E1 = SQRT(E1 / FLOAT(N))
  WRITE(6,55) E1,ANF,T
55 FORMAT(4F15.5,E2,F2.6,15X,3F15.5,F2.5,6X,13HTIME CONSTANT E1:5)
   IF(KKK.FC.,2) CC TC 56
   KKK = 2
   G = RNS / E1
   CC TC 58
57 CC 5S I = 1,NT
   BETA(J,J) = BETA(J,J) * G
50 E1 = E1 + BETA(J,J)*2
```
CC TO 56
56 CONTINUE
C
C AKC = 8.33E+0
C CELP = HEAT OF REACTION  12000 BTU/LBMOL
C AKF = 14.000
C TRI = TEMPERATURE OF THE REACTANTS  175.  F
C
223 READ(5,15) AKC,CELF,AKA,TRI
C
C RP = SET POINT
C CAC = INITIAL CONCENTRATION OF REACTANT A
C CA = CONCENTRATION OF REACTANT A LEAVING REACTOR
C KF = FLOW RATE OF REACTANTS
C TFC = TEMPERATURE OF THE PRODUCTS LEAVING REACTOR
C KI = MANIPULATIVE VARIABLE COOLANT FLOW RATE
C TKI = INLET TEMPERATURE OF COOLANT
C TMC = TEMPERATURE OF COOLANT LEAVING REACTOR
C DELT = INTEGRATION TIME IN SECONDS
C
10 FORMAT(4E13.5)
Y(1) = 0.
XP(1) = 0.
PT = 216.86
KZ = C
SW = 73.5
SAC = .5975
STWI = 83.
STRI = T1
PCA = .2369
PTPC = 213.86
PC = 5.
SWI = 24.99
WI = SWI
PW = SWI
PTWC = 235.21
DELT = 1.
K = SW

C
C
C = SCAC
TKI = SIKI
TRI = STRI
CC 26C 1 = 1,22.
IF(1, EQ. 2., ) WI = SW1 + 10.
IF(I, EQ. 12.) WI = SW1
IF(1, AE. 1.) CC 10, 111
WI = S8I
IF(1, GT. 50.) WI = SW1 + 10.
111 CC 23C J = 1,1.
7C CC 26C JJ = 1,3
APTPC = PTPC + 456.6G
RA = (AKC / 64.) * (XP(-AKA / AFTPO))
TFC = (k*(TRI-PTFC)/44154. + (CELFT / 49.5) * RA = PCA*+2
1 = -.1+.24G2 * (PTPO - PTPC)) * CELT + PTPC
CA = (k*(CAC - PCA)/44154. - RA * PCA*+2) * CELT + PCA
TK2 = (k*(TAC - PTWC)/32348.16 + -5769 * (PTPO - PTWC))
1 = *CELTC + PTWC
PTPC = TPC
PCA = CA1
2C3 PTWC = TwC
XP(1 ) = WI - PK
YF(1+1) = TFC - FT
PK = WI
26C WRITE(6,252) k, CAC, TRI, TKI, WI, TPC, CA, RA, TK2
252 FORMAT(1X, 3E12.5)
| RETURN
| END

C        INPUT DATA
| 4  4
0010203055 40. 50. 56 1.
| 104.  193.  1.  55.  25.  4.  706
APPENDIX E

COMPUTER PROGRAM

FOR

PREDICTIVE CONTROL

OF A

SECOND ORDER SYSTEM
C  PREDICTIVE CONTROL OF SECOND ORDER SYSTEM
C
C DIMENSION P(5),A(7)
C CALL FPTRAP(*1)
20 READ(5,6) NPASS
9 FORMAT(11,5)
C IF(NPASS .EQ. C) CALL EXIT
C
C P = SIX PARAMETERS AS DETERMINED BY OPTIMIZATION METHOD
C READ(5,15) (P(I), I = 1,6)
C 1C FORMAT(6F15.5)
C IC = F(1)
C C IC
C C A = CONTROLLER PARAMETERS
C A(1) = (P(1) - C) / (C + 1. - P(1))
C A(2) = 1. / (A(2) * (1. - P(2) + P(3)) * (Q + 1. - P(1)))
C A(3) = P(2) * A(2)
C A(4) = P(3) * A(2)
C WRITE(6,30) (A(I), I = 1,4)
3C FORMAT(4F15.5)
C XP1 = 0.
C XP2 = 0.
C PWV = 0.
C VP = 0.
C SVF = 0.
C SFP = 0.
C RP = 0.
C L = C
C 5C I = 1,NPASS
C CALL FCOC(VF,CP)
C C XP1   CALCULATED AT THIS POINT TO ALLOW FOR MAXIMUM AND MINIMUM
C C VALUE OF MANIPULATED VARIABLE
C XP1 = (VP + A(1)*PWV + A(3)*XP2 - A(4)*XP3)/A(2)
C SVF = SVF + VP+PF
\begin{align*}
PV &= V_P \\
FP &= CP - RP \\
SFP &= SFP + FP \\
SFP_2 &= SFP_2 + FP \times 2 \\
XP &= (P(5) + C \times F(4)) \times FP - P(6) \times SFP \\
C &= \text{MANIPULATED VARIABLE:} \\
VM &= -A(1) \times PV + A(2) \times XP - A(3) \times XP_1 + A(4) \times XP_2 \\
XP_3 &= XP_2 \\
XP_2 &= XP_1 \\
XP_1 &= XP \\
50 &= \text{CONTINUE} \\
C &= \text{SUM OF VALUE MOVEMENT SQUARED} \\
&= \text{WRITE}(6,121) \text{ SYF} \\
121 &= \text{FOLDER}(6:\text{VALUE F3G.5}) \\
&= \text{GC TO 2.} \\
1111 &= \text{STDP} \\
&= \text{END} \\
\end{align*}

SUBROUTINE PREC(VX,CP) 
CPYXCA L, KPASS 
DIMENSION YA(150),R(150),BETA(150),AB(6),VM(5) 
L = L + 1 
IF(L,NE,1) \text{CC TC 1} \\
C &= \text{GENERATE NOISE FOR OUTPUT} \\
XC &= + 84 \\
YC &= + 63 \\
ZP &= \text{RAND}(XC,YC) \\
XC &= YC \\
C &= \text{STEP INFLUENCE HEIGHT} \\
C &= \text{ANC = CUTLET NOISE SA} \\
C &= \text{ANN = UNMEASURABLE INLET NOISE FACTOR} \\
C &= \text{ANN = UNMEASURABLE INLET NOISE SCALE FACTOR} \\
C &= \text{ANC = UNMEASURABLE CUTLET NOISE SCALE FACTOR} \\
C &= \text{AMP = MEASURED CUTLET NOISE SCALE FACTOR}
C

T = TIME CONSTANT OF FIRST ORDER LAG USED TO GENERATE NOISE

RMS = DESIRED ROOT MEAN SQUARE OF NOISE

READ(5,4) PG,RA1,RA2,ANP,T,RMS

4 FORMAT(6F10.5)

NPASS = 2

NPASS1 = NPASS + 1

CC 3 I = 1,NPASS1

R(I) = RAND(XC,YC) * ANP * 2. - ANP

3 IF(I .EQ. 1) BETA(I) = R(I)

KKK = 1

C = 1.

53 NT = 5

E1 = C.

CA = C.

CT = .2

IF(I .EQ. 2) CC TC 57

CC 55 I = 1,NPASS

BETA(I) = CA

FLAG = R(I)

CR = (R(I+1) - R(I)) / FLOAT(NT)

CC 51 J = 1,NT

CA = CA + (G * FLAG - CA) / T * CT

51 FLAG = FLAG + CR

56 E1 = E1 + BETA(I)**2

59 E1 = SQRT(E1/FLOAT(NPASS))

WRITE(6,55) E1,ANP,T

55 FORMAT(7H,62PLT,F2.9,JX,3HANP,F10.5,13HTIME CONSTANT,F10.5)

IF(KKK .EQ. 2) CC TC 56

KKK = 2

G = RMS / E1

CC TC 53

57 CC 5E I = 1,NPASS

BETA(I) = BETA(I) * G

58 E1 = E1 + BETA(I)**2
C 1 Dynamic parameters of system
C 2 Dead time in sampling periods
C 3 Parameters related to time constants of
C 4 System
C 5 System gain

56 READ (E,1) (AB(I), I = 1,4), NC
1 FORMAT (4F13.5,11c)
   XP = .
   SFP = .
   XIM1 = .
   XI = .
   YA(I) = .
   CC = .
   SCP = .
   SUCP = .
   UCP = .
   J = AE(1)
   FJ = J
10 CC 100 I = 1,NC
   C = .
   VM(L) = VX + C
   VX = .
   K = L - J
   IF(K) 19,19,2:
20 IF(K-1) 30,30,4:
40 FLAG = (FJ+1.-AE(I))*VM(K) + (AP(I)-FJ)*VM(K-1)
   GC TC 6:
19 FLAG = .
   GC TC 6.
30 FLAG = (FJ+1.-AF(I)) * VM(K)
   60 XIPL = AB(2)*XI-AP(2)*XIM1+AB(4)*A(1,-AB(2)+AB(3))*FLAG
   XIM1 = XI
   XI = XIP1
X_P = X_P + X_1P1
Z_P = RAND(XC,YC) * ARC + 2. - ARC + BITA(L)
Y_A(L+1) = X_P + Z_P
C_P = Y_A(L+1)
C_C = C_C + Q
U_C_P = BETA(L) + C_C
WRITE(6,2)L,VM(L),CF,U_C_P
2 FORMAT(1X,I5,1X,5HP = F20.5,1C2,5HC_P = F20.5,5X,6HUC_P = F20.5)
S_C_P = S_C_P + C_P**2
SUC_P = SUC_P + U_C_P**2
IF(L.EQ.2) WRITE(6,8) S_C_P,SUC_P
100 L = L + 1
8 FORMAT(17D) CONTROLLED ERROR F20.5/19HOUNCONTROLLED ERROR F20.5)
L = L - 1
RETURN
END

INPUT DATA
2CD
0.4 1.5 0.5625 1.25 1.6732
1.8
0.4 1.5 0.5625 1.25 1
APPENDIX F

COMPUTER PROGRAM

FOR

PREDICTIVE CONTROL

OF A

CHEMICAL

REACTOR
C   PREDICTIVE CONTROL OF CHEMICAL REACTOR

COMMON L, NPASS
DIMENSION P(6), A(7)
KKLM = 0
CALL FPTRAP(+1)
20 READ(5,6) NPASS
9 FORMAT(113)
   KKLM = KKLM + 1
   IF(NPASS .EQ. 0) CALL EXIT
C   P = SIX PARAMETERS AS DETERMINED BY OPTIMIZATION METHOD
   READ(5,13) (P(I), I = 1, 6)
13 FORMAT(6F1), 5)
   IC = F(1)
   C = IC
C   A = CONTROLLER PARAMETERS
   A(1) = (P(1) - C) / ( C + P(1) - P(1))
   A(2) = 1. / (P(4) + (1. - P(2) + P(3)) * (Q + 1. - P(1)))
   A(3) = P(2) * A(2)
   A(4) = P(3) * A(2)
22 WRITE(6,38) (A(I), I = 1, 4)
30 FORMAT(4F15, 5)
   XP1 = C.
   XP2 = 0.
   XP3 = 0.
   PVW = 0.
   VM = 0.
   SVW = 0.
   SFP = 0.
   RP = 218.86
   SFP2 = 0.
   L = 0.
   CC 5: I = 1,NPASS
   CALL FREQ(VF, CF)
C   XF1 CALCULATED AT THIS POINT TO ALLOW FOR MAXIMUM AND MINIMUM
C   VALUE OF MANIPULATED VARIABLE
XP1 = (VM + A(1) * VM + A(3) * XP2 - A(4) * XP1) / A(2)
SVM = SVM + VM >= 2
PVV = VM
FP = CP - RP
SP = SP + FP
SP2 = SP2 + FP >= 2
XP = -(P(5) + CP * P(6)) * FP - P(6) * SP 
C   VM = MANIPULATED VARIABLE
VM = -A(1) * PVV + A(2) * XP - A(2) * XP1 + A(4) * XP2
XP3 = XP2
XP2 = XP1
XP1 = XP
50 CONTINUE
C   SVV = SUM OF VALUE MOVEMENT SQUARED
C   SP2 = SUM OF ERROR FROM SET POINT SQUARED
WRITE(6, 121) SVM, SP2
121 FORMAT(6F15.5) "CONTROLLS ERROR E2: 5"
CC TO 2.
1111 STOP
END

CCTCCA L, NPASS
DIMENSION BETA(4, 256), R(256)
L = L + 1
IF (L .NE. 1) CC TO 2;
C   GENERATE NOISE FOR REACTOR FEED STREAMS
C   RMS = ROOT MEAN SQUARE VALUE OF NOISE
CC 5E J = 1,4
C   XA AND YA NUMBERS TO INITIALIZE RANDOM NUMBER GENERATOR
C   AAM = 1.
C   T = TIME CONSTANT OF FIRST ORDER LAG USED TO GENERATE NOISE
C   RMS = DESIRED ROOT MEAN SQUARE VALUE OF NOISE
C PC = STEP INLET HEIGHT
C N = NUMBER OF RANDOM NUMBERS TO GENERATE
REAL(5,4) XC, YC, ANM, T, RMS, PC, A
4 FORMAT(6F12.5, 11.)
ZP = RAND(XC, YC)
XC = YC
KI = N * 10
NIP = KI + 1
CC 31 = 1, NIP
R(I) = RAND(XC, YC) * ANM + 7. - ANM
3 IF(T, EC, 3.) BETA(J, I) = R(I)
K KK = 1
G = 1.
53 AT = 5
E1 = G.
CA = G.
CT = .2
IF(T, EC, 3.) CC TO 57
CC, 5C I = 1, N
BETA(J, I) = CA
FLAG = R(I)
CR = (R(I+1) - R(I)) / FLOAT(AT)
CC 51 LL = 1, N1
CA = CA + (G*FLAG - CA) / T * CT
51 FLAG = FLAG + G8
50 E1 = E1 + BETA(J, I)*G2
59 E1 = SCRT(E1 / FLCM(N))
WRITE(6, 55) E1, AAM, I
55 FORMAT(4H1HIMS, 6F2.8, 1X, 3EAM, F10.5, 9X; 13!!! TIME CONSTANT F1: .5)
IF(KKK, EC, 2) CC TO 56
K KK = 2
G = RMS / E1
CC TO 53
57 CC 5E 1 = 1, N
BETA(J, I) = BETA(J, I) * G
58 E1 = E1 + BETA(J, I) * 2
59 CONTINUE
   A = 8.33645
   CELH = HEAT OF REACTION 121000 BTU/LBMOLE
   AKA = 14.000
   TRI = TEMPERATURE OF THE REACTANTS 175 F
   READ(5, 10) AKC, CELH, AKA, TRI
10 FORMAT(4E10.5)
   RP = SET POINT
   CAC = INITIAL CONCENTRATION OF REACTANT A
   CA = CONCENTRATION OF REACTANT A LEAVING REACTOR
   W = MASS FLOW RATE OF REACTANTS
   TEF = TEMPERATURE OF THE PRODUCTS LEAVING REACTOR
   WI = MANIPULATIVE VARIABLE COOLANT FLOW RATE
   TI = INLET TEMPERATURE OF COOLANT
   TAC = TEMPERATURE OF COOLANT LEAVING REACTOR
   CELT = INTEGRATION TIME IN SECONDS
   Sk = 73.5
   SCAC = .3975
   STWI = 8.
   STRI = TRI
   PCA = .2069
   PTAC = 218.86
   SWI = 26.89
   WI = SWI
   Ph = SWI
   PTMC = 225.31
   CFLT = 10.
   PWI = WI
   kI = WI + VX
   IF(WI .LT. 1.) WI = 1.
   IF(WI .GT. 200.) WI = 200.
Brian Ramaker is the son of Mr. and Mrs. Shirley Ramaker of Kohler, Wisconsin. He was born in Sheboygan, Wisconsin, on April 5, 1941. He attended elementary and secondary schools in Kohler, Wisconsin, graduating from Kohler High School in June, 1959.

His undergraduate work was done at the University of Wisconsin, graduating with a Bachelor of Science Degree in Chemical Engineering in June, 1963. He received his Master of Science Degree in Chemical Engineering from the University of Wisconsin in August, 1964.

In August, 1964 he married the former Judith Gasser of Kohler, Wisconsin. He was employed full-time by the Ethyl Corporation in Baton Rouge, Louisiana, from September, 1964 until September, 1966. On December 30, 1965 his wife made him the proud father of a daughter, Ruth Marie. In September, 1966 he entered Louisiana State University as a full-time graduate student. He is now the candidate for the degree of Doctor of Philosophy in the Department of Chemical Engineering.
EXAMINATION AND THESIS REPORT

Candidate: Brian Lee Ramaker

Major Field: Chemical Engineering

Title of Thesis: Stochastic Control of Chemical Processes

Approved:

[Signatures]

Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

[Signatures]

Date of Examination:

July 17, 1968