FINAL REPORT
Grant Number NAG-1-1378

EFFECT OF DESIGN SELECTION ON RESPONSE SURFACE PERFORMANCE

from

William C. Carpenter
Department of Civil Engineering and Mechanics
University of South Florida
Tampa, Florida 33620

to

Jean-Francois M. Barthelemy
Office of Interdisciplinary Research
Structural Dynamics Division
NASA Langley Research Center
Hampton, Virginia 23681-0001
# Table of Contents

1. **Introduction** ................................................................. 1  
   1.1 **Quality of Fit** ......................................................... 2  
      1.1.1 Fit at the designs .................................................. 2  
      1.1.2 Overall fit ......................................................... 3  
   1.2 **Polynomial Approximations** ........................................... 4  
      1.2.1 Exactly-determined approximation ............................... 5  
      1.2.2 Over-determined approximation .................................. 5  
      1.2.3 Under-determined approximation ................................ 6  
   1.3 **Artificial Neural Nets** ............................................... 7  
2. **Levels of Designs** ........................................................... 10  
   2.1 **Taylor Series Approximation** ........................................ 10  
   2.2 **Example** ............................................................... 13  
   2.3 **Conclusion** ............................................................ 18  
3. **Standard Designs** ........................................................... 19  
   3.1 **Underlying Principle** ................................................ 19  
   3.2 **Statistical Concepts** ................................................ 20  
   3.3 **Orthogonal Designs** ................................................ 23  
      3.3.1 **Scaling** .......................................................... 24  
         3.3.1.1 Example of Scaled Designs: ................................. 25
3.3.2 Bias ......................................................... 26

3.3.2.1 A bias example--linear approximating polynomial but the exact function contains linear terms and cross-product terms: ................................................. 27

3.3.2.2 A bias example--linear approximating function but the exact function is a complete quadratic polynomial: .... 28

3.3.3 Orthogonal Designs for Linear Approximations ............ 29

3.3.4 Orthogonal Designs for 2nd Order Polynomial Approximations .......................................................... 30

3.3.5 General Discussion of Orthogonal Designs .................. 31

3.4 Central Composite Designs--Designs for Fitting Second Order Models .... 31

3.4.1 Format of the central composite design ..................... 31

3.4.1.1 Single center point rotatable second order experimental designs: .................................................. 32

3.4.1.2 Multiple center point rotatable uniform precision designs: ......................................................... 33

3.4.1.3 Single center point orthogonal central composite designs: ......................................................... 34

3.4.1.4 Rotatable orthogonal designs: ................................. 35

3.4.2 Discussion of the central composite design .................. 36

3.4.3 Example--Fox's Banana Function ............................ 37

3.4.4 Conclusion ............................................... 44
7. Additional Examples .................................................. 67
   7.1 Introduction .................................................. 67
   7.2 The 35 Bar Truss with 4 Design Variables ...................... 67
   7.2 The 35 bar truss with 15 design variables ...................... 71
   7.3 Analytical function--20 design variables ...................... 74
   7.4 Conclusion .................................................. 76

8. Augmented Minimum Point Designs .................................... 77
   8.1 Introduction .................................................. 77
   8.2 Augmented Minimum Point Designs for 2nd Order Approximations 79
       8.2.1 Specifics of program DESIGNS ............................ 80
   8.3 Augmented Minimum Point Design for 3rd and 4th Order
       Approximation .................................................. 81
       8.3.1 Specifics of program DESIGN4 ............................ 82
   8.4 Conclusion .................................................. 83

9. Solution Algorithm .................................................. 84
   9.1 Introduction .................................................. 84
   9.2 Program Specifics .............................................. 84

10. Conclusion ...................................................... 86

11. References ...................................................... 89
1. Introduction

The mathematical formulation of the engineering optimization problem is

\[ \min f(x) \]
\[ \text{subject to } g_i(x) \leq 0, \quad i = 1, q \]  \hspace{1cm} (1)

where

\(\{x\}\) is an nx1 matrix of design variables,

\(f(\{x\})\) is the objective function, and

\(g_i(\{x\})\) are constraint equations.

Evaluation of the objective function and constraint equations in Equation (1) can be very expensive in a computational sense. Thus, it is desirable to use as few evaluations as possible in obtaining its solution. In solving Equation (1), one approach is to develop approximations to the objective function and/or restraint equations and then to solve Equation (1) using these approximations in place of the original functions. These approximations are referred to as response surfaces.

The desirability of using response surfaces depends upon the number of functional evaluations required to build the response surfaces compared to the number required in the direct solution of Equation (1) without approximations. The present study is concerned with evaluating the performance of response surfaces so that a decision can be made as to their effectiveness in optimization applications. In particular, this study focuses on how the
quality of approximations is effected by design selection. Polynomial approximations and neural net approximations are considered.

To provide the groundwork for future discussion, this introductory section discusses:

1. measures of quality of fit at the designs and measures of quality of fit over a region of interest and
2. the methodology used to build the approximations.

1.1 Quality of Fit

Let us consider a problem with \( n \) design variables, the components of the vector \( \{x\} = \{x_1, x_2, \ldots, x_n\}^t \). A total of \( N \) designs will be considered: \( \{x\}_j, j = 1,N \). At the designs \( \{x\}_j \), let

\[
y_j = \text{the value of the function to be approximated}
\]

\[
\hat{y}_j = \text{the value of the approximating function.}
\]

The approximating function, \( \hat{y} \), should closely match the function, \( y \), not only at the designs, \( \{x\}_j \), but over the entire region of interest.

1.1.1 Fit at the designs

The approximating function \( \hat{y} \) closely approximates the function \( y \) when \( s \) is small where

\[
s = \sqrt{\frac{\delta^2}{N}} \tag{2}
\]

and where \( \delta^2 \) is the sum of the squares of the residuals thus
\[ \delta^2 = \sum_{i=1}^{N} (y_i - \bar{y})^2 \]  \hspace{1cm} (3)

Let \( \bar{y} \) be the average value of the designs, \( y_i \). Thus

\[ \bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i \]  \hspace{1cm} (4)

In this study, one measure of the closeness of fit to be considered is the non-dimensional value \( v \) where

\[ v = \left( \frac{\sum_{i=1}^{N} (y_i - \bar{y})^2}{\bar{y}^2 \frac{1}{N}} \right) \times 100 \]  \hspace{1cm} (5)

The coefficient \( v \) is the non-dimensional root mean square (RMS) error at the designs. Thus, \( v = 0 \) is a necessary and sufficient condition that the approximating function fit the actual function at the \( N \) design points.

### 1.1.2 Overall fit

Just because the approximating function exactly fits the function at \( N \) designs does not guarantee that it gives a good fit over the region of interest. It is therefore desirable over the region of interest to have a measure of the quality of overall fit. Several examples of this study considers a two dimensional region of interest. For these problems, the
rectangular region of interest is overlaid with a 31x31 evenly spaced grid of points. The value of the function and the approximating function is then compared at these NG = 961 evenly spaced grid of points. Other examples consider a rectangular n dimensional region of interest. These regions of interest are also overlaid with a evenly spaced grid of points. The value of the function and the approximating function are then compared at these NG grid points. For these examples, a measure of the quality of overall fit is taken as

\[ v_G = \sqrt{\frac{\sum_{i=1}^{NG} (y_i - \bar{y})^2}{NG}} \]  \times 100 \]  

where \( \bar{y}_G \) is the average value of y at the grid points. A small value of \( v_G \) indicates that the approximating function did a good job of approximation over the region of interest.

1.2. Polynomial Approximations

With the polynomial response surface approach, the approximating function is taken as an \( m = k + 1 \) term polynomial expression [1-3] thus

\[ \hat{y} = b_0 + b_1 x_1 + \ldots b_k x_k \]  

where \( x_i \) is some expression involving the design variables. For example, a second order polynomial approximation in two variables could be of the form
\[
y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_1^2 + b_4 x_1 x_2 + b_5 x_2^2
\]  
(8)

The value of the function to be approximated at the N designs can be used to determine the \( m = k + 1 \) undetermined coefficients in the polynomial expression. For the N designs, Equation (7) yields

\[
\begin{bmatrix}
y_1 \\
y_2 \\
\vdots \\
y_N
\end{bmatrix} =
\begin{bmatrix}
1 & X_{11} & \cdots & X_{1k} \\
1 & X_{21} & \cdots & X_{2k} \\
\vdots & \vdots & \ddots & \vdots \\
1 & X_{N1} & \cdots & X_{Nk}
\end{bmatrix}
\begin{bmatrix}
b_0 \\
b_1 \\
\vdots \\
b_k
\end{bmatrix}
\]  
(9)

or

\[
[Y] = [Z][b]  
\]  
(10)

where \( \{Y\} \) is an Nx1 matrix, \( [Z] \) is an Nxm matrix, and \( \{b\} \) is an mx1 matrix.

1.2.1 Exactly-determined approximation

When \( N = m \), the approximation is exactly-determined and the matrix \( \{b\} \) can be determined from Equation (10).

1.2.2 Over-determined approximation

With \( N > m \), Equation (10) can be solved in a least squares sense thus [1-3]
Equation (12) in effect, chooses the terms of \( \{b\} \) so as to minimize the square of the residual as defined in Equation (2).

### 1.2.3 Under-determined approximation

When \( N < m \), the approximation is under-determined. A solution can be obtained by choosing the terms of \( \{b\} \) so as to minimize the square of the residual as defined in Equation (2). However, a direct solution can be obtained by using the concept of pseudo-inverse [4,5]. Assume that the rank of matrix \( [Z] \) is \( N \) and define the pseudo-inverse of matrix \( Z \), \( Z^* \) thus

\[
[Z]^* = [Z]'([Z][Z]')^{-1}
\]  

(13)

where \( t \) denotes transpose. Solution of Equation (10) is then

\[
\{b\} = [Z]^*\{Y\} + [Q]\{w\}
\]

(14)
where \( \{w\} \) is an \((m-N)\) column matrix of arbitrary coefficients and \([Q]\) is a \(mx(m-N)\) matrix formed from any \(m-N\) independent columns of the matrix \([R]\) thus

\[
[R] = [I] - [Z]'[Z] 
\]  

One solution to Equation (14) is to take all the arbitrary terms of \( \{w\} \) as zero giving

\[
\{b\} = [Z]'\{Y\} 
\]  

The basic solution to Equation (10) is Equation (16). Using that equation, at the designs, \( \{x\}_j \), the value of \( \hat{y}_j \) matches the value of \( y_j \). If \( w_i \) is the \( i \)th term in matrix \( \{w\} \) and \( \{q\}_i \) is the \( i \)th column of matrix \([Q]\), then at the designs, \( \{x\}_j \), \( \hat{y}_j = 0 \) when

\[
\{b\} = w_i\{q\}_i 
\]  

Thus, the last term of the right hand side of Equation (14) gives \( \hat{y}_j \) values which match \( y_j \) at the designs, \( \{x\}_j \), for any values of \( w_i \).

1.3 Artificial Neural Nets

While the initial motivation for developing artificial neural nets was to develop computer models that could imitate certain brain functions, neural nets can be thought of as another way of developing a response surface. Different types of neural nets are available [6,7], but the type of neural nets considered in this paper are back propagation nets with one hidden layer as shown in Figure 1. This type of neural net has been used previously to develop
response surfaces [8-12] and is capable, with enough nodes on the hidden layer, of approximating any continuous function [13].

For the neural net of Figure 1, associated with each node on the hidden layer, node j, and each output node, node k, are coefficients or weights, \( \theta_j \) and \( \theta_k \), respectively. These weights are referred to as the biases. Associated with each path, from an input node i to node j on the hidden layer, is an associated weight, \( w_{ij} \) and from node j on the hidden layer to output node k is an associated weight \( w_{jk} \). Let \( q_i \) be inputs entered at node i. Node j on the hidden layer receives weighted inputs, \( w_{ij}q_i \). It sums these inputs and uses an activation function to yield an output \( r_j \). The activation function considered in this paper is the sigmoid function [6,7]

\[
    r_j = \frac{1}{1 + e^{-\sum w_{ij}q_i - \theta_j}}
\]  

(18)

Output node k then receives inputs \( w_{jk}r_j \) which are summed and used with an activation function to yield an output \( s_k \). Some variation of the delta-error back propagation algorithm [6,7] is then used to adjust the weights on each learning try so as to reduce the values between the predicted and desired outputs. In this investigation, studies were performed using the program NEWNET [14] which was developed especially for this investigation. NEWNET minimizes the sum of the squares of the residuals in Equation (2) with respect to the weights and biases of the net. Training of the net is thus formulated as an unconstrained minimization problem. Solution of this minimization problem is performed
using the method of Davidon, Fletcher, and Powell [15-16]. That algorithm performs a series of one dimensional searches along search directions. Search directions are determined by building an approximation to the inverse Hessian matrix using gradient information. Gradients required by that algorithm are obtained using back-propagation. One-dimensional searches are performed along the search directions using an interval shortening routine.
2. Levels of Designs

2.1 Taylor Series Approximation

The overriding factor which affects the accuracy of an approximation is the levels of the design parameters considered. It is instructive to consider a problem in two design variables. Suppose we wish to make a quadratic approximation of a function thus:

\[ y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_1^2 + b_4 x_1 x_2 + b_5 x_2^2 \ldots \]  

(19)

Consider that the exact function is evaluated at 6 design points and the information thus generated will be used to determine the 6 undetermined coefficients in Equation (19). Design variables at these design points are taken from the following sets:

\[ x_1 \text{ from the set } \{x_{11}, x_{12}, \ldots, x_{1p}\} \]
\[ x_2 \text{ from the set } \{x_{21}, x_{22}, \ldots, x_{2q}\} \]

(20)

Here \( p \) discrete values are considered for \( x_1 \) and \( q \) discrete values are considered for \( x_2 \). The variable \( x_1 \) is said to have \( p \) levels and \( x_2 \) is said to have \( q \) levels. The problem is to determine the minimum levels of the design variables, \( p \) and \( q \), required to build the quadratic approximation. In this regard, it is instructive to consider a Taylor series approximation [17] of the function about the point \( \{x_1 = 0, x_2 = 0\} \):

\[ \hat{y} = y(0,0) + \{\nabla y(0,0)\}^T \Delta x + \{\nabla^2 y(0,0)\}^T [H(0,0)]^T \Delta x + \ldots \]  

(21)
where

\[
\{ \Delta x \} = [(x_1 - 0) \ (x_2 - 0)]' = [x_1 \ x_2]'
\]  \hspace{1cm} (22)

\[
\{ v(0,0) \} = [(\frac{\partial y(0,0)}{\partial x_1}) \ \frac{\partial y(0,0)}{\partial x_2}]'
\]  \hspace{1cm} (23)

\[
[H(0,0)] = \begin{bmatrix}
\frac{\partial^2 y(0,0)}{\partial x_1^2} & \frac{\partial^2 y(0,0)}{\partial x_1 \partial x_2} \\
\frac{\partial^2 y(0,0)}{\partial x_1 \partial x_2} & \frac{\partial^2 y(0,0)}{\partial x_2^2}
\end{bmatrix}
\]  \hspace{1cm} (24)

Entering Equations (22), (23), and (24) into Equation (21) gives

\[
y = y(0,0) + \frac{\partial y(0,0)}{\partial x_1} x_1 + \frac{\partial y(0,0)}{\partial x_2} x_2 + \frac{\partial^2 y(0,0)}{\partial x_1^2} x_1^2 + 2 \frac{\partial^2 y(0,0)}{\partial x_1 \partial x_2} x_1 x_2 + \frac{\partial^2 y(0,0)}{\partial x_2^2} x_2^2
\]  \hspace{1cm} (25)

The derivatives in Equation (25) can be determined by finite difference equations [18]. The second derivative of \( y \) with respect to \( x_1 \) can be obtained using information at points indicated in Figure 2 by solid circles, the second derivative of \( y \) with respect to \( x_2 \) can be
obtained using information at points indicated by unfilled circles, and the mixed derivative can be obtained using information at points indicated by unfilled squares.

It can be seen in Figure 2 that at least three levels of both $x_1$ and $x_2$ must be used to obtain a quadratic approximation. If three levels are not provided, not information is available to calculate the higher derivatives in Equation (25). A complete 3 factorial design does not have to be used -- only 6 selected points from the complete 3 factorial design. Information at those 6 points allow the undetermined coefficients to be exactly determined.

Consider now the design of Figure 3 which are also taken from the 3 factorial design. Even though 6 design points are used, this set of design points does not allow an approximation containing the $x_2^2$ term of Equation (25). However, with the design of Figure 3, an approximation of the form of Equation (26) could be obtained thus:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_1^2 + b_4x_1x_2$$

(26)

With the design of Figure 3, if a solution is attempted using Equations (19) and (12), a singular coefficient matrix will be encountered. A solution could be attempted using the pseudo-inverse concept of Equations (13) and (14). However, recent studies [19] have shown that non-unique solutions are obtained with this technique. Non-uniqueness makes these solutions undesirable. Using Equations (26) and (12), a slightly over-determined approximation is obtained.
Recent studies have found that the numerical performance of neural network approximations and polynomial approximations with the same number of associated undetermined parameters is comparable [19]. Thus, it is not expected that neural nets as approximators will perform better than polynomials when there are inadequacies in the training design, as in Figure 3. The next example investigates performance of both polynomial and neural net approximations.

2.2 Example

Consider the function

\[ y = 1 + x_1 + x_2 + x_3 + x_1^2 + x_1 x_2 + x_1 x_3 + x_2^2 + x_2 x_3 + x_3^2 \]  \hspace{1cm} (27)

In the first phase of the investigation, approximations are to be made of this function using the design of Figure 4. The star pattern of design points in Figure 4 does not allow mixed derivatives of the function to be calculated using finite difference type formulae but does permit the other second derivatives to be calculated. Thus, information is available to make a polynomial approximation of the form

\[ \hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_1^2 + b_5 x_2^2 + b_6 x_3^2 \]  \hspace{1cm} (28)

The function \( y \) was evaluated at the design points shown in Figure 4 yielding 7 training pairs for calculating the 7 undetermined parameters in Equation (28). The value of the approximating function \( \hat{y} \) was then evaluated at a 5x5x5 grid of designs. These values of \( \hat{y} \)
were then used to evaluate $v_G$ from Equation (6). The value of $v_G$ obtained is shown in the
first line of Table 2.1.

Table 2.1. Performance of Approximations for Various Designs

<table>
<thead>
<tr>
<th>Number Designs Points</th>
<th>Description</th>
<th>Polynomial Approximation</th>
<th>Neuronal Net Approximation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No. Para.</td>
<td>$v_G$ (%)</td>
</tr>
<tr>
<td>7</td>
<td>Star--see Figure 4</td>
<td>7</td>
<td>34.6</td>
</tr>
<tr>
<td>12</td>
<td>Star--see Figure 5</td>
<td>7</td>
<td>34.6</td>
</tr>
<tr>
<td>10</td>
<td>Computer Generated</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>3 factorial</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>125</td>
<td>5 factorial</td>
<td>10</td>
<td>0.0</td>
</tr>
</tbody>
</table>

A neural net approximation was then considered. Previous studies [19] have indicated that it is desirable to have more training pairs than the number of undetermined parameters (weights and biases) associated with the net. If fewer training pairs than undetermined parameters are used, non-unique approximations should be expected. For a neural net with one hidden layer as shown in Figure 1, there are 6 parameters associated with a net with one node on the hidden layer and 11 parameters associated with a net with two nodes on the hidden layer. It was considered that one node on the hidden layer would yield an inadequate approximation. Thus 2 nodes on the hidden layer were considered. Thus, the
neural net approximation is under-determined. That is to say that there are fewer training pairs than there are undetermined parameters associated with the approximation. Non-unique approximations are to be expected. Indeed, this was the case. The 8 training pairs were used to make 10 different approximations by having training commence from a different randomly selected set of weights and biases. Once the nets were trained, the value of the approximating function, $\hat{y}$, was generated at the $5\times 5\times 5$ set one grid points and the value of $v_G$ was developed. The range of the values obtained is shown in Table 2.1. One can see that a large range of values is obtained. The best neural net approximation is only slightly better than the polynomial approximation while the worst neural net approximation is considerably worse. Just as with the polynomial approximation, the designs used to train the approximation can not yield information necessary to capture essential features of the function to be approximated.

The 12 designs of Figure 5 were next used in the training of a polynomial approximation and a 2 node neural net approximation. Even though more designs are used here than in Figure 4, the additional designs selected do not yield any more information about the nature of the function being approximated. Information is still not available for determining the mixed derivatives of the function to be approximated. Thus, the polynomial approximation of Equation (26) was considered. As there are now more training pairs than there are undetermined parameters, the approximation obtained is over-determined. As no new information is available with the 12 designs, the same polynomial approximation and thus
the same $v_G$ as before are obtained. The value of $v_G$ is shown in the second line of Table 2.1.

A neural net with 2 nodes on the hidden layer was then trained with the 12 training pairs. The net was trained 10 times starting from different randomly selected sets of weights and biases. Even thought the number of training pairs, 12, is greater than the number of undetermined parameters associated with the net, 11, non-unique approximations were obtained as can be seen in Table 2.1. Thus, it can be concluded that for neural net approximations, having more training pairs than the number of associated undetermined parameters is only a necessary condition for obtaining a unique approximation but that it is not a sufficient condition. As the 12 designs offered no new information about the function being approximated over that offered by the 8 designs, then just as with the 8 design case, non-unique approximations were obtained.

The program DESIGNS [20], which was developed for this project, was used to generate 10 designs which contain the information necessary for calculating the 10 undetermined coefficients of the complete quadratic approximation of the form:

$$
\hat{y} = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3 + b_4 x_1^2 + b_5 x_2^2 + b_6 x_3^2 + b_7 x_1 x_2 + b_8 x_1 x_3 + b_9 x_2 x_3
$$

The location of these design points is shown in Figure 6. The polynomial approximation obtained by training the polynomial of Equation (29) with the computer generated designs exactly duplicated the test function of Equation (27). Thus, $v_G$ for the 5x5x5 grid of points
was zero as seen in the third line of Table 2.1.

A neural net with 2 nodes on the hidden layer with 6 associated undetermined parameters and a neural net with 3 nodes on the hidden layer and 11 associated undetermined parameters were then trained 10 times with the computer generated training pairs. Each training started from a different randomly selected set of weights and biases. For the case of 2 nodes on the hidden layer, the approximation generated was over-determined and a unique approximation was obtained (the small range of \( v_G \) obtained most likely results from the exit criteria employed in the training algorithm). For the case of 3 nodes on the hidden layer, there are 11 associated undetermined parameters but only 10 training pairs. Thus the approximation is under-determined and a non unique approximation is obtained as can be seen in Table 2.1.

The performance of the neural net approximations was much poorer than that of the polynomial approximation on this problem. This poorer performance may be in part because the problem is biased towards the polynomial approximation as the function being approximated is 2 second order polynomial.

A complete \( 3^3 \) factorial design and a \( 5^3 \) factorial design were considered to see if good results could be obtained with the neural nets if more training pairs were employed. Indeed this was the case. However, many more training pairs were required to get a good approximation than were required with the polynomial approximation. The extra training
pairs were wasted on the polynomial approximation. Ten correctly selected training pairs is all that is required to get an exact second order approximation. The additional training pairs offered no new information to the polynomial approximation. The coefficient $v_G$ was zero for training pairs using the 3 and 5 factorial designs and a second order polynomial approximation.

2.3 Conclusion

For a given order of approximation, a good design must use an adequate number of levels of the design variables or a poor approximation will be obtained. Likewise, design points must be located so that information is available for determining all of the undetermined coefficients of the approximating function. In many instances, especially when the region of interest is small, a second order polynomial approximation or neural net equivalent will be sufficient to build a response surface. A second order approximation requires a design containing 3 levels of the design variables. Program DESIGNS has been developed to generate a minimum point design which allows all of the coefficients of a second order polynomial approximating function to be obtained. This minimum point design can be augmented by randomly selected design points or by user selected points.
3. Standard Designs

3.1 Underlying Principle

When making a polynomial approximation of a function, the number of design levels required for each design variable depends upon the order of polynomial approximation being used. Consider for example the problem of approximating a function y, a function of one design variable. As previously discussed, two levels of the design variable would be required to make a linear approximation of the function, three levels of the design variable would be required to make a second order approximation, four levels of the design variable would be required to make a 3rd order approximation, etc. If y is a function of r design variables, a pth order polynomial approximation, \( y \), requires designs at \( p+1 \) levels in each design variable.

In response surface methodology, the term factor is used for design variable. A factorial design or factorial experiment is a design in which one uses each of the possible combinations of the levels of each factor. If \( m \) is the number of level of each factor and \( r \) is the number of factors, then the design would be referred to as a \( m^r \) factorial experiment. Table 3.1 gives the number of designs in various factorial experiments.
Table 3.1. Number of designs in a full factorial design

<table>
<thead>
<tr>
<th>m = level</th>
<th>r = factor</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>27</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>81</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
<td>59049</td>
<td>1.05E06</td>
<td></td>
</tr>
</tbody>
</table>

One can see that even for a small number of factors, complete factorial experiments become impractical if designs are computationally or experimentally expensive to obtain. One then is forced to use some sub-set of the factorial design or alternate designs containing requiring fewer design points. Concepts from statistics are normally used in selecting a sub-set of the factorial design or in developing alternate designs. Thus statistical concepts are reviewed.

3.2 Statistical Concepts

When making an approximation, \( \hat{y} \), of a function, \( y \), most approaches used to select design points for a design consider that

1. polynomial approximations are employed and

2. the value of the function, \( y_p \), determined at the designs, \( \{x\}_p \), contains some error, \( \epsilon_p \).
A measure of the error at point $i$ is the variance of the error, $\text{var}(\epsilon_i) = \sigma^2$ where

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mu)^2$$

(30)

where

- $\mu$ is the true mean of all possible observations of $y_i$ and
- $n$ is the number of observations made.

In experimental investigations, $\epsilon_i$ is experimental error. When making approximations to analytical functions, $\epsilon_i$ is zero and the variance of the error at point $i$ is zero. Often approximations are made to a function whose values must be obtained from some numerical algorithm such as the finite element method or finite difference method. Values of $y_i$ obtained from such algorithms depend on control parameters which dictate the level of accuracy of the solution. For example, if $y$ was a stress determined from a finite element analysis, then $y$ could depend on a control parameter which specifies the coarseness of the finite element idealization. In this case, different values of $y_i$ would be obtained for the $i$th design for different values of the control parameters and $\epsilon_i$ could be thought of as a numerical error.

It would be an interesting study to select designs such that approximations developed are insensitive to numerical errors such as finite element idealization error. However, the problem at hand is to find a good approximation to an analytical function or a good
approximation for output from a deterministic model. For the problem at hand, for a given design, $x_i$, one obtains the same functional value, $y_i$, no matter how many times the function is evaluated. Thus, the problems considered in this report contain no numerical error. However, as all known algorithms with one exception [21] consider that there is some experimental or numerical error, this section now further examines this case.

Errors in the value of $y_i$ used to build an approximation affect the estimation of the undetermined coefficients, $b_j$, in the polynomial approximation and thus affect $\hat{y}_i$, the values of $y_i$ predicted by the approximation. A measure of the error in $b_j$ resulting from errors in $y_i$ is the variance of $b_j$. For example, consider that $y_i$ is obtained from a finite element analysis and that a $p$th order polynomial approximation is employed. The undetermined coefficients in that approximations, $b_j$, can be determined from Equation (12). If a number of approximations were now made with finite element results, obtained using different idealizations, the coefficient $b_j$ for these approximations would be different. The variance of $b_j$ is a measure of how much the $b$'s change for these different approximations. In like form, the different approximations yield different $\hat{y}_i$ and the variance of $\hat{y}_i$ is a measure of how much the $\hat{y}_i$ values change from approximation to approximation.

From a numerical standpoint, it is desirable to have approximations that are not highly sensitive to the error $\epsilon_i$. Approximations are insensitive to the error, $\epsilon_i$, if the variance of $b_j$ and the variance of $\hat{y}_i$ is small. Most design selection algorithms currently in use attempt in some way to keep these variances small.
The variance of $b_j$ is the $j,j$ term of the variance-covariance matrix $\text{cov} \, b$ where (see Equation 3.11 of [3] or Equation 2.8 of [2])

$$[\text{cov} \, b] = \sigma^2 ([Z]'[Z])^{-1} \tag{31}$$

and the variance of $\hat{y}_i$ is given by (see Equation 2.11 of [2])

$$\text{var} \, \hat{y}_i = \sigma^2 \{Z_i\}'([Z]'[Z])^{-1} \{Z_i\} \tag{32}$$

where $\{Z_i\}'$ is the $1 \times p$ vector whose elements correspond to the elements of a row of matrix $[Z]$.

Notice that these variance involve the matrix $[H]$ where

$$[H] = ([Z]'[Z])^{-1} \tag{33}$$

Design selection affects $[Z]$, which from Equation (33) affects $[H]$, which in turn affects the variances of $b_j$ and $\hat{y}_i$. Many design point selection algorithms attempt to select designs which give an $[H]$ matrix which will keep the variances of $b_j$ and $\hat{y}_i$ small.

### 3.3 Orthogonal Designs

The associated undetermined coefficients of a polynomial approximation function can be found from Equation (12). The solution for these coefficients involve the matrix $[Z]$ (see Equations (9) and (10)). Let $\{Z_i\}$ be the $i$th column of matrix $[Z]$. A design is said to be
orthogonal if the columns of the $[Z]$ matrix are orthogonal, i.e. $\{Z_i\}^T[Z_j] = 0, \; i \neq j$. There are interesting properties of orthogonal designs which have prompted their use. Thus orthogonal designs will now be presented in some detail.

### 3.3.1 Scaling

The discussion of orthogonality is simplified by working with scaled variables. Consider that the approximation in question involves $k$ unscaled design variables $\bar{x}_i$ and contains $N$ design points. Instead of working with $\bar{x}_i$, the variables will be scaled. Let $\bar{x}_{iu}$ be the $u$th level of unscaled variable $i$ and $x_{iu}$ be the scaled level. The desired scaling is

$$\sum_{u=1}^{N} x_{iu}^2 = N, \; i=1,k \quad (34)$$

$$\sum_{u=1}^{N} x_{iu} = 0, \; i=1,k \quad (35)$$

This scaling can be accomplished by having

$$x_{iu} = \frac{\bar{x}_{iu} - \bar{x}_i}{S_i} \quad (36)$$

where

$$\bar{x}_i = \text{the average of the levels of } \bar{x}_i \quad (37)$$

and
With this scaling, $N$ experimental design points of the orthogonal design give

$$S_i^2 = \sum_{u=1}^{N} \frac{(x_{iu} - \bar{x}_i)^2}{N} \quad (38)$$

$$[Z]'[Z] = N[I] \quad (39)$$

$$([Z]'[Z])^{-1} = \frac{1}{N}[I] \quad (40)$$

where $[I]$ is the identity matrix.

### 3.3.1.1 Example of Scaled Designs:

Consider a 2 factorial design with levels of 4 and -4. For that design

$$\bar{x}_1 = 0, \quad \bar{x}_2 = 0 \quad (41)$$

and

$$S_1^2 = S_2^2 = \frac{(4-0)^2 + (-4-0)^2}{2}, \quad \text{or} \quad S_1 = S_2 = 4 \quad (42)$$

From Equation (3), the levels of the scaled variables are

$$x_{iu} = \frac{x_{iu} - 0}{4} \quad (43)$$

or the levels of the scaled variables are 1 and -1.
3.3.2 Bias

Assume that the polynomial approximating function is inadequate. The coefficients of that polynomial can be determined from Equation (12). Let \( \{ \hat{b}_1 \} \) be the coefficients thus obtained and let \( [Z_1] \) be the corresponding \([Z]\) matrix. Then from Equation (12)

\[
\{ \hat{b}_1 \} = ([Z_1]'[Z_1])^{-1}[Z_1]'[Y]
\] (44)

Assume that the function being approximated can be expressed as

\[
{Y} = [Z]{b}
\] (45)

where

\[
{b} = \begin{bmatrix} {b}_1 \\ {b}_2 \end{bmatrix}, \quad [Z] = [ \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} ]
\] (46)

Entering Equations (40), (45), and (46) into Equation (44) gives

\[
\{ \hat{b}_1 \} = \frac{1}{N}[I][Z_1]'[[Z_1] \quad [Z_2]]\begin{bmatrix} {b}_1 \\ {b}_2 \end{bmatrix}
\] (47)

Entering Equation (39) into Equation (47) gives

\[
\{ \hat{b}_1 \} = \frac{1}{N}(N[I]{b}_1 + [Z_1]'[Z_2]{b}_2)
\] (48)

or

26
where \([A]\) is called the alias matrix. One can see in Equation (49) that the coefficients \(\{\hat{b}_1\}\) will only be correct estimates of \(\{b_1\}\) if the columns of \([Z_1]\) are orthogonal to the columns of \([Z_2]\). Special situations where this orthogonality occurs are next discussed.

3.3.2.1 A bias example--linear approximating polynomial but the exact function contains linear terms and cross-product terms:

Consider a linear approximating polynomial

\[
\hat{y} = \hat{b}_o + \sum_{i=1}^{k} \hat{b}_i x_i
\]  

(50)

where the exact function is

\[
y = b_o + \sum_{i=1}^{k} b_i x_i + \sum_{i=1}^{k} \sum_{j=i}^{k} b_{ij} x_i x_j
\]  

(51)

where \(b_{ij}\) are the undetermined coefficients associated with the cross-product terms. For this problem, a full \(2^k\) factorial design gives that the columns of \([Z_1]\) are orthogonal to the columns of \([Z_2]\) and thus
\{\hat{b}_1\} = \{b_1\}

3.3.2.2 A bias example--linear approximating function but the exact function is a complete quadratic polynomial:

Consider a linear approximating polynomial

\[ \hat{y} = \hat{b}_0 + \sum_{i=1}^{k} \hat{b}_i x_i \]  

where the exact function is a complete second order polynomial thus

\[ y = b_0 + \sum_{i=1}^{k} b_{ii} x_i^2 + \sum_{j=1}^{k} \sum_{j=1}^{k} b_{ij} x_i x_j \]  

Assume again that a full $2^k$ factorial design is used. For this problem the alias matrix is such that one obtains

\[ \hat{b}_o = b_o + \sum_{i=1}^{k} b_u \]
\[ \hat{b}_j = b_p, \quad j=1,k \]

Thus only $\hat{b}_o$ is biased with the other coefficients unbiased or uncorrelated.
3.3.3 Orthogonal Designs for Linear Approximations

For a problem with \( r \) design variables, a full \( 2^r \) factorial design is an orthogonal design if the approximating function is a first order polynomial. There are several advantages in using such an orthogonal design when the approximating function is assumed to be linear. These advantages are:

1. The solution for the coefficients of the polynomial approximation require a matrix inverse (see Equation (12)). However, when the design is an orthogonal design, that inverse is very easily obtained using Equation (40). Thus there is a small computational advantage in using an orthogonal design.

2. Examples 3.3.2.1 and 3.3.2.2 indicate that under certain conditions, the coefficients obtained using an orthogonal design are unbiased. Obtaining unbiased coefficients is probably more important in developing response surface from experimental results than when developing response surfaces when results are from a deterministic model. With experimental studies, it may be important to ascertain the unbiased values of the linear coefficients. For the deterministic model however, one is looking for an approximating function which gives a good approximation throughout a region of interest. Whether the coefficients of the polynomial approximation are biased or unbiased is of little concern.

3. It can be proven that for linear polynomial approximations, an orthogonal design gives the minimum variance of the coefficients (see page 109 of [3]). It is important when modeling experimental results to obtain a model that is not overly sensitive to experimental error and thus there is an advantage in having a minimum variance of the coefficients.
However, for response surfaces of a deterministic model, variance of the coefficients is not relevant.

3.3.4 Orthogonal Designs for 2nd Order Polynomial Approximations

It is not possible to find an orthogonal design when using a second order polynomial approximating function of the form of Equation (8) (see page 107 of \[2\]). However, an orthogonal design can be found if one uses as the approximating function a second order orthogonal polynomial (page 130 of \[3\])

\[
y = b_0 + \sum_{i=1}^{k} b_i x_i + \sum_{i=1}^{k} b_{ii}(x_i^2 - \bar{x}_i^2) + \sum_{i=1}^{k} \sum_{j=1}^{k} b_{ij} x_i x_j
\]  

(56)

where

\[
\bar{x}_j = \frac{\sum_{u=1}^{N} x_{j,u}^2}{N}
\]

(57)

and where

\[N = \text{the number of design points and}
\]
\[x_{j,u} = x_j \text{ for each of the design points.}\]

(58)

The use of an orthogonal design still gives the small computational advantage that the inverse shown in Equation (12) is an inverse of a diagonal matrix. However, when using
second order approximations, it is not clear under what conditions one obtains unbiased coefficients. Also it can not be proven that orthogonal designs any longer give a minimum variance of the coefficients. Thus most of the reasons for using orthogonal designs found for linear approximations are not present when using second order approximations.

3.3.5 General Discussion of Orthogonal Designs

Orthogonal designs offer a small computational advantage that the matrix inverse required in solving for the coefficients of the polynomial approximating function is an inverse of a diagonal matrix. When approximating a deterministic model, properties of orthogonal designs which minimize the variance of the coefficients and which give unbiased coefficients are unimportant. For this case, the use of orthogonal designs can only be justified by how well they perform on test problems. Such test problems are presented later in this report.

3.4 Central Composite Designs--Designs for Fitting Second Order Models

It was shown in Section 2 that at least 3 levels of the design variables are required if one is to make a second order approximation. A workable alternative to using a $3^k$ factorial design is a class of designs called the central composite design. These types of designs are widely used by workers applying second order response surface techniques [3].

3.4.1 Format of the central composite design

The central composite design is a design composed of the $2^k$ factorial design augmented by additional points. The augmented design points are as follows:
Figure 7 shows a central composite design for $k=3$. The value of $\alpha$ and the number of design points at the center of the design are varied to meet certain conditions. In the following, those conditions are chosen assuming that the approximating polynomial function is given by Equation (56).

### 3.4.1.1 Single center point rotatable second order experimental designs:

A design is said to be rotatable when the variance of the estimated response—that is, the variance of $\hat{y}$, which in general is a function of position in the design space, is instead only a function of the distance from the center of the design and not on the direction. In other words, a rotatable design is one for which the quality of the estimator $\hat{y}$ is the same for two points that are the same distance from the center of the design [3]. It is possible to develop central composite designs which have a single center point. The value of $\alpha$ which will yield these rotatable second order designs are given in Table 3.2.
Table 3.2. Value of $\alpha$ for single center point rotatable central composite designs

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.414</td>
</tr>
<tr>
<td>3</td>
<td>1.682</td>
</tr>
<tr>
<td>4</td>
<td>2.000</td>
</tr>
<tr>
<td>5</td>
<td>2.378</td>
</tr>
<tr>
<td>5 (1/2 rep)</td>
<td>2.000</td>
</tr>
<tr>
<td>6</td>
<td>2.828</td>
</tr>
<tr>
<td>6 (1/2 rep)</td>
<td>2.378</td>
</tr>
<tr>
<td>7</td>
<td>3.364</td>
</tr>
<tr>
<td>7 (1/2 rep)</td>
<td>2.828</td>
</tr>
<tr>
<td>8</td>
<td>4.000</td>
</tr>
<tr>
<td>8 (1/2 rep)</td>
<td>3.364</td>
</tr>
</tbody>
</table>

Note in Table 3.2 that a rotatable second order experimental design can be obtained with a fractional factorial design augmented with additional design points as well as with a augmented full factorial design.

3.4.1.2 Multiple center point rotatable uniform precision designs:

In general, the variance of $\hat{y}$ varies with distance from the center of the design. However, by varying the number of center points, $N$, the variance at a distance of unity from the center can be made approximately equal to the variance at the center of the design. Such designs are referred to as **uniform precision designs**. The uniform precision design is based on the philosophy that in the central region of the design space there should be uniform importance as far as the variance of response is concerned, as opposed to, for example, a
situation in which the variance is low in the center of the design but increases drastically as one moves away from the design center [3]. The number of center points, m, and the value of α can be varied so as to obtain a rotatable uniform precision designs. Table 3.3 gives those values.

Table 3.3. Values of m and α for multiple center point rotatable uniform precision designs

<table>
<thead>
<tr>
<th>k</th>
<th>m</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>1.414</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>1.682</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>2.000</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>2.378</td>
</tr>
<tr>
<td>5 (1/2 rep)</td>
<td>6</td>
<td>2.000</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>2.828</td>
</tr>
<tr>
<td>6 (1/2 rep)</td>
<td>9</td>
<td>2.378</td>
</tr>
<tr>
<td>7 (1/2 rep)</td>
<td>14</td>
<td>2.828</td>
</tr>
<tr>
<td>8 (1/2 rep)</td>
<td>20</td>
<td>3.364</td>
</tr>
</tbody>
</table>

3.4.1.3 Single center point orthogonal central composite designs:

An orthogonal central composite design can be developed where \([Z]'[Z]\) is diagonal. To obtain a design of this type a single center point can be used and the α value are taken from Table 3.4.
Table 3.4. Values of $\alpha$ for single center point orthogonal central composite designs

<table>
<thead>
<tr>
<th>k</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.000</td>
</tr>
<tr>
<td>3</td>
<td>1.216</td>
</tr>
<tr>
<td>4</td>
<td>1.414</td>
</tr>
<tr>
<td>5</td>
<td>1.596</td>
</tr>
<tr>
<td>6</td>
<td>1.761</td>
</tr>
<tr>
<td>7</td>
<td>1.910</td>
</tr>
<tr>
<td>8</td>
<td>2.045</td>
</tr>
</tbody>
</table>

3.4.1.4 Rotatable orthogonal designs:

By varying the number of designs at the design center, $m$, and by selecting appropriate values for $\alpha$, an orthogonal rotatable central composite design can be obtained. Values of $m$ and $\alpha$ for such a design are given in Table 3.5.
Table 3.5. The value of m and α for multiple center point orthogonal rotatable central composite designs

<table>
<thead>
<tr>
<th>k</th>
<th>m</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8</td>
<td>1.414</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>1.682</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>2.000</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>2.378</td>
</tr>
<tr>
<td>5 (1/2 rep)</td>
<td>10</td>
<td>2.000</td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>2.828</td>
</tr>
<tr>
<td>6 (1/2 rep)</td>
<td>15</td>
<td>2.378</td>
</tr>
<tr>
<td>7 (1/2 rep)</td>
<td>22</td>
<td>2.828</td>
</tr>
<tr>
<td>8 (1/2 rep)</td>
<td>33</td>
<td>3.364</td>
</tr>
</tbody>
</table>

3.4.2 Discussion of the central composite design

Orthogonal central composite designs have been shown to give a variance of response comparable to that obtained with a full $3^k$ factorial design. Thus, their use is justified when one has experimental error in the response function. Rotatable and uniform precision designs attempt to control the response variance. Thus there use is also justified when one has experimental error in the response function. However, when building a response surface for a deterministic model where there is no experimental error in the response function, their use is justified only by how well they perform of trial problems. Likewise, the designs were developed for the approximating function of Equation (56). If a different second order polynomial approximating function such as in Equation (8) were used or if a neural net was used to develop the response surface, then again the justification for the use of the various
central composite designs would have to be based on their performance on trial problems. Performance of various central composite designs on trial problems is next reported.

3.4.3 Example -- Fox's Banana Function

Fox investigated in Reference [16] a function

\[ y = 10x_1^4 - 20x_2x_1^2 + 10x_2^2 + x_1^2 - 2x_1 + 5 \] (60)

which has banana shaped contours as seen in Figure 8. The region of interest to be considered is (-1.5 < x_1 < 1.5, -.5 < x_2 < 2.0).

A second order polynomial approximation is to be made of this function using an orthogonal polynomial approximation as in Equation (56). A two variable orthogonal polynomial approximation is of the form

\[ \hat{y} = b_0 + b_1x_1 + b_2x_2 + b_{11}(x_1^2 - \bar{x}_1^2) + b_{22}(x_2^2 - \bar{x}_2^2) + b_{12}x_1x_2 \] (61)

where

\[ \bar{x}_j = \frac{\sum_{k=1}^{N} x_{j,k}^2}{N} \] (62)

and where
In the first phase of this example, Fox's function was approximated using the second order orthogonal polynomial of Equation (61). The designs used in making the approximation were:

1. a full $5^2$ factorial design,
2. a full $3^2$ factorial design,
3. single center point rotatable central composite design,
4. multiple center point rotatable uniform precision central composite design,
5. single center point orthogonal central composite design,
6. multiple center point rotatable orthogonal central composite design,
7. minimum point design from program DESIGNS,
8-10. minimum point design from program DESIGNS augmented by additional randomly selected design points, and
11. nine randomly selected design points.

Once an approximation was obtained, the approximate function was evaluated at a $31 \times 31$ grid of points over the region of interest. The approximate function values at these 961 points were used to develop the error parameter $v_G$ from Equation (6). Because there are a differing number of functional evaluations required for each of the sundry designs tested, a comparison of the designs based on $v_G$ is misleading. For example, the full $5^2$ factorial
design has 25 design points each requiring a functional evaluation where as the multiple center point rotatable orthogonal central composite design has but 16 design points requiring 9 functional evaluations (in the following it is assumed that the function being approximated has no experimental or numerical error and thus the 8 design points at the design center require but one functional evaluation). Thus a comparison of performance based only on quality of fit is not a fair comparison. The $5^2$ factorial might do a better job of approximating a function but the computational cost of the $25-9=16$ extra functional evaluations might make it a less desirable design.

For each design, design $j$, a measure of efficiency, $E_j$, was developed where

$$E_j = \frac{(v_G)_{design_j} T_{design_j}}{(v_G)_{design_1} T_{design_1}}$$

(64)

where $T$ is the number of functional evaluations required for a given design.

The efficiency of all the designs was compared to design 1, the $5^2$ factorial design. Table 3.6 gives, for each design tested, the number of design points, $N$; for central composite designs, the number of design points at the center of the design, $m$; the number of functional evaluations required, $T$; the value of $v$; the value of $v_G$; and the value of $E_j$. 

39
Table 3.6. Performance of various designs on Fox's Banana Function, orthogonal polynomial approximating function, -1.5 \(< x_1 < 1.5, \ -0.5 < x_2 < 2.0

<table>
<thead>
<tr>
<th>Design</th>
<th>N</th>
<th>m</th>
<th>T</th>
<th>(v)</th>
<th>(v_G)</th>
<th>(E_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(^2) factorial design</td>
<td>25</td>
<td>...</td>
<td>25</td>
<td>70.76</td>
<td>78.92</td>
<td>1.00</td>
</tr>
<tr>
<td>3(^2) factorial design</td>
<td>9</td>
<td>...</td>
<td>9</td>
<td>64.07</td>
<td>102.46</td>
<td>.47</td>
</tr>
<tr>
<td>single center point rotatable central composite design</td>
<td>9</td>
<td>1</td>
<td>9</td>
<td>54.36</td>
<td>77.34</td>
<td>.35</td>
</tr>
<tr>
<td>multiple center point rotatable uniform precision central composite design</td>
<td>13</td>
<td>5</td>
<td>9</td>
<td>53.08</td>
<td>77.34</td>
<td>.35</td>
</tr>
<tr>
<td>single center point orthogonal central composite design</td>
<td>9</td>
<td>1</td>
<td>9</td>
<td>64.07</td>
<td>102.46</td>
<td>.47</td>
</tr>
<tr>
<td>multiple center point rotatable orthogonal central composite design</td>
<td>16</td>
<td>8</td>
<td>9</td>
<td>51.62</td>
<td>77.34</td>
<td>.35</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS</td>
<td>6</td>
<td>...</td>
<td>6</td>
<td>0</td>
<td>162.62</td>
<td>.49</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS augmented by 2 randomly selected design points</td>
<td>8</td>
<td>...</td>
<td>8</td>
<td>43.27</td>
<td>105.16</td>
<td>.43</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS augmented by 3 randomly selected design points</td>
<td>9</td>
<td>...</td>
<td>9</td>
<td>53.53</td>
<td>88.63</td>
<td>.40</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS augmented by 4 randomly selected design points</td>
<td>10</td>
<td>...</td>
<td>10</td>
<td>53.05</td>
<td>86.44</td>
<td>.44</td>
</tr>
<tr>
<td>random--9 points</td>
<td>9</td>
<td>...</td>
<td>9</td>
<td>21.05</td>
<td>460.96</td>
<td>2.10</td>
</tr>
</tbody>
</table>
Several items can be noted in Table 3.6:

1. The design composed of 9 randomly selected design points did poorly. Even though the design points were chosen randomly, it turned out that the design points were not well scattered in the design space but were heavily concentrated in one quadrant of the design space. The polynomial approximation fitted the function well at the design points but poorly over the region of interest.

2. The value of $v_6$ was approximately the same for the single center point rotatable central composite design, the multiple center point rotatable uniform precision central composite design, and the multiple center point rotatable orthogonal central composite design. These three designs differ only in the number of design points at the center of the design space. These designs have 1, 5, and 8 designs at the center, respectively. The effect of putting more designs at the center is to translate the response surface toward the center response. For this problem, however, the actual and approximated response were very close at the design center point, even for only 1 design point at the center. Thus, adding more design points at the design center did little to translate the response surface and thus did not material effect the value of $v_6$.

3. The eleven designs of Table 3.5 were next used to build an approximation using the standard second order polynomial approximation of Equation (8) instead of the orthogonal polynomial approximation of Equation (61). Results identical to those of Table 3.5 were found. The type of approximating polynomial may effect variances but does not affect quality of fit at the design points or over the region of interest. For those problems were there is no experimental or numerical error associated with functional evaluations, one is
not interested in variance. Thus, there is little advantage in using the orthogonal polynomial approximating functions over a standard second order polynomial function.

4. Based on efficiency, the single center point rotatable central composite design, the rotatable uniform precision central composite design, and the rotatable orthogonal central composite design performed the best but none of the designs gave a good approximation over the region of interest. Over a small region of interest, one could expect that a second order polynomial approximation could well approximate the given function. Obviously, here the region of interest is too large for a second order approximation to be a good one. Thus a smaller region of interest was chosen, \(-0.5 < x_1, 0.5, -0.5 < x_2 < 0.5\). Table 3.7 compares the eleven designs using this region of interest. Notice that over this smaller region of interest, all the designs gave a much better approximation to the function.

5. For the smaller region of interest, based on efficiency, the \(3^2\) factorial design, the single center point orthogonal central composite design, and the augmented minimum point designs performed the best. Obviously, the optimum choice of design is problem dependent. However, all designs except the randomly selected design performed much better than the \(5^2\) factorial design.
Table 3.7. Performance of various designs on Fox's Banana Function, orthogonal polynomial approximating function, \(-0.5 < x_1 < 0.5, -0.5 < x_2 < 0.5\)

<table>
<thead>
<tr>
<th>Design</th>
<th>N</th>
<th>m</th>
<th>T</th>
<th>v</th>
<th>v_G</th>
<th>E_j</th>
</tr>
</thead>
<tbody>
<tr>
<td>$5^2$ factorial design</td>
<td>25</td>
<td>...</td>
<td>25</td>
<td>11.16</td>
<td>8.57</td>
<td>1.00</td>
</tr>
<tr>
<td>$3^2$ factorial design</td>
<td>9</td>
<td>...</td>
<td>9</td>
<td>13.27</td>
<td>10.95</td>
<td>.46</td>
</tr>
<tr>
<td>single center point rotatable central composite design</td>
<td>9</td>
<td>1</td>
<td>9</td>
<td>6.58</td>
<td>14.74</td>
<td>.62</td>
</tr>
<tr>
<td>multiple center point rotatable uniform precision central composite design</td>
<td>13</td>
<td>5</td>
<td>9</td>
<td>5.88</td>
<td>14.74</td>
<td>.62</td>
</tr>
<tr>
<td>single center point orthogonal central composite design</td>
<td>9</td>
<td>1</td>
<td>9</td>
<td>13.27</td>
<td>10.95</td>
<td>.46</td>
</tr>
<tr>
<td>multiple center point rotatable orthogonal central composite design</td>
<td>16</td>
<td>8</td>
<td>9</td>
<td>5.47</td>
<td>14.74</td>
<td>.62</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS</td>
<td>6</td>
<td>...</td>
<td>6</td>
<td>0</td>
<td>18.66</td>
<td>.52</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS augmented by 2 randomly selected design points</td>
<td>8</td>
<td>...</td>
<td>8</td>
<td>5.74</td>
<td>11.82</td>
<td>.44</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS augmented by 3 randomly selected design points</td>
<td>9</td>
<td>...</td>
<td>9</td>
<td>6.45</td>
<td>10.53</td>
<td>.44</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS augmented by 4 randomly selected design points</td>
<td>10</td>
<td>...</td>
<td>10</td>
<td>6.33</td>
<td>10.29</td>
<td>.48</td>
</tr>
<tr>
<td>random--9 points</td>
<td>9</td>
<td>...</td>
<td>9</td>
<td>2.42</td>
<td>47.22</td>
<td>1.98</td>
</tr>
</tbody>
</table>
3.4.4 Conclusion

Second order polynomial approximations or neural net equivalents are often adequate for building response surfaces, especially if the region of interest is small. Central composite designs are convenient for building the second order approximations. They provide the necessary information for determining all of the coefficients of the approximating polynomial and give a good distribution of points in the design space. The approximating function can be made to closely fit the exact function at the design center by using multiple center points. When modeling deterministic systems, each functional evaluation at the design center yields the same function value. Thus, for deterministic models, only one functional evaluation need be performed at the center point even when multiple center points are used. Table 3.8 gives information relevant to central composite designs for various number of design variables, k. Central composite designs give over-determined second order polynomial approximations. In other words, there are more design points in the design than there are undetermined coefficients in a second order polynomial approximation. Table 3.8 also gives the percentage that the approximation is over-determined. Previous studies [19] have indicated that designs which give approximations that are around 20-50% over-determined tend to be efficient designs. One can see that the central composite designs are reasonable for k<6. For larger k values, too many design points are being used by the central composite designs. For k>5, an augmented minimum point design is a better choice.
Table 3.8. Information relevant to central composite designs for various number of design variables

<table>
<thead>
<tr>
<th>Number of design variables, k</th>
<th>Number of coefficients in a 2nd order polynomial approximation</th>
<th>Number of functional evaluations required with a central composite design</th>
<th>% over-determined</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>33</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>8</td>
<td>33</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>14</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>24</td>
<td>60</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>42</td>
<td>50</td>
</tr>
<tr>
<td>6</td>
<td>28</td>
<td>76</td>
<td>171</td>
</tr>
<tr>
<td>7</td>
<td>36</td>
<td>142</td>
<td>294</td>
</tr>
<tr>
<td>8</td>
<td>45</td>
<td>272</td>
<td>504</td>
</tr>
</tbody>
</table>
4. Optimality Criteria

4.1 D, A, E, G, and V Optimality Criteria

It was pointed out in Section 3 that even for a small number of factors, a complete factorial experiment become impractical if functional evaluations are computationally or experimentally expensive to obtain and thus one is forced to use some subset of the factorial design or an alternate design requiring fewer experiments. Section 3 shows that the variances of the coefficients of a polynomial approximation and the variance of the predicted response involve the matrix \([H]\) given in Equation (33) and repeated here:

\[
[H] = ([Z]'[Z])^{-1}
\]  

(65)

Schoofs [22] lists five criteria for selecting a subset of the factorial designs. These criteria involve the matrix \([H]\). The criteria, referred to as optimality criteria, attempt to make \([H]\) minimal. However, "the minimum of a matrix is not a well defined concept and a number of operational criteria have been developed" [22]. The optimality criteria for selecting a subset of a full factorial design can be based on selecting the subset satisfying the following criteria:

1. D-optimality, which is achieved if the determinant of \([H]\) is minimal which in term gives that the product of the eigenvalues of \([H]\) is minimal.

2. A-optimality, which is achieved if the trace of \([H]\) is minimal which in term gives that the sum of the eigenvalues of \([H]\) is minimal.

3. E-optimality, which is achieved if the largest eigenvalue of \([H]\) is minimal.
4. G-optimality, which is achieved if the maximum over all candidate points of the estimated response variance is minimal.

5. V-optimality, which is achieved if the estimated response variance, averaged over all candidate points is minimal.

4.1.1 Criteria Applied to a One Dimensional Example

An example is considered here to compare the performance of the 5 optimality criteria. The following test function of one variable was considered:

\[ y = 2 + x + \sin \left[ \frac{3\pi}{2} (x + 1) \right], \quad -1 \leq x \leq 1 \] (66)

This function was approximated with polynomials of order 1-4. The approximations shown in Figure 9 were developed using 13 designs, uniformly spaced in the region of interest. These approximations were then used to generate the functional values at 61 uniformly spaced points in the region of interest which were used to plot the curves of Figure 9.

Further approximations of Equation (66) were developed using various number of design points, n. The designs selected were

1. uniformly spaced design points, n = 5, 7, 9, 11, 13;
2. randomly selected design points, n = 5, 7, 8, 11, 13;
3. an n member subset of the 13 uniformly spaced design points, n = 5, 7, 9, 11.
Under item 3, the subset of design points was chosen using:

1. D-optimality,
2. A-optimality,
3. E-optimality,
4. G-optimality, and
5. V-optimality.

A FORTRAN program was written to perform the investigation under item 3. The demanding part of the programming was to identify all the possible subsets from the set of thirteen design points. After developing a procedure to identify all combinations, each subset was used to build the [H] matrix. The "optimal" [H] matrix was then determined using the five optimality criteria. The coefficient \( v_G \) was then computed for the optimal subset. Figures 10-13 show the value of \( v_G \) for the D, A, E, and G optimality criteria when a first, second, third, and fourth order approximation is being made, respectively, versus the number of design points specified in the subset. Also shown in those figures is the value of \( v_G \) for designs consisting of design points uniformly spaced in the region of interest.

It was found that for all subsets of size \( r \) from a design point set of size \( n \) that the estimated response variance, averaged over all candidate points, was invariant. This finding undoubtedly could also be proven theoretically but such a proof was not attempted. From this example, one can conclude that the V optimality criteria, which employs the estimated average response variance, is not a viable criteria for selecting a subset of design points from
a given set. From Figures 10-13, one can see that in most cases there is little difference in
the performance of the various optimality criteria with criteria D and G performing slightly
better than the other two criteria. As can be seen in Figure 12, on one occasion (when
using a third order polynomial approximation and when selecting a subset of 5 design points
from the 13 design point set) the G optimality criteria performed poorly while the D criteria
did not. Thus, this example indicates that the D optimality criteria may be the criteria of
choice. There is a further advantage in using the D optimality criteria. The requirement
that the determinant of $[H]$ is minimal is equivalent to a requirement that the determinant
of $[G]$ is maximal where

$$[G] = [Z][Z] \quad (67)$$

Thus the D optimality criteria insures that the procedure for determining polynomial
coefficients in Equation (12) will be well defined. In other words, Equation (12) uses the
inverse of $[G]$. The D optimality criteria guarantees that $[G]$ is not singular.

One can see in Figures 10-13 that, in most cases, all the optimality criteria performed worst
than the uniformly spaced design case. This example indicates that a design picked using
an optimality criteria may be no better than a design of the same size in which the design
points are uniformly located in the design space.

### 4.2 S and O Optimality Criteria

The previous optimality criteria involved only the matrix $[H]$ and did not consider the
function to be approximated. Thus for a given number of design variables and level of approximation, the same designs would be selected no matter what the nature of the function to be approximated. Initially it was thought that a superior optimality criteria would have to consider the nature of the function. Thus two additional optimality criteria were examined:

1. S-optimality, which is achieved if the average error of approximation at the design points is minimal and

2. Q-optimality, which is achieved if the maximum error of approximation at the design points is minimal.

Here

\[
\text{average error of approximation} = \frac{\sum_{i=1}^{r}(y_i - \hat{y}_i)^2}{r} \quad (68)
\]

and

\[
\text{maximum error of approximation} = \max_{i=1,..,r} (y_i - \hat{y}_i)^2 \quad (69)
\]

where \( r \) is the size of the subset of design points to be selected. One can see that with the S and Q optimality criteria, the function to be approximated effects the design points selected.
4.2.2 Criteria Applied to a One Dimensional Example

The one dimensional example problem of Section 4.1.1 was then re-examined. Figures 14-17 show values of $v_G$ using the S and Q optimality criteria and using a first, second, third, and fourth order polynomial approximation, respectively, versus size of the subset of design points. Also shown in these figures are results for uniformly spaced design points. One can see in these figures that terrible approximations were obtained with these criteria when only small subsets of design points were selected from the original set. Figures 18-20 indicate why such bad approximations are obtained with these two criteria.

Figure 18 depicts results obtained by having eleven designs points selected, using the Q optimality criteria, from a set of 13 design points. The Q optimality criteria finds an approximation such that the maximum error of the approximation over eleven design points is minimal. One can see in Figure 18 that the approximating function did indeed well fit the exact function at the 11 design points selected. However, the approximating function did a poor job of approximation at the ends of the region of interest and thus would not yield a low value of $v_G$. Figure 19 is similar to Figure 18 except that this figure depicts results obtained by having 7 design points selected from the set of 13 design points. One can see that for the optimum design selected, there is an almost perfect approximation at the design points selected but over a much larger region the approximation is poor and thus a large value of $v_G$ would be obtained. In Figure 20, only 5 design points are selected. Again at those design points, an almost perfect approximation is obtained but a terrible approximation is obtained over a large part of the region of interest and thus a large $v_G$
would be obtained. Thus we can conclude that the S and O optimality criteria are not operative.

4.3 An Alternate Approach--Random Selection of Designs

The effect of randomly picking design points was next considered. Here designs are picked in the region of interest using a random number generator.

4.3.1 Random Selection of Designs Applied to a One Dimensional Example

For the one dimensional problem under consideration, first, second, third, and fourth order approximations were considered. Design point sets containing 5, 7, 9, 11, and 13 design points were developed by randomly picking design points in the region of interest using a random number generator. Approximations were developed using the design sets. Results using these approximations are compared in Figures 21-24 to results using uniformly spaced design points. One can see in these figures that most of the time results from randomly picked design points are either as good as or not much worst than results from uniformly spaced design points. However, on two occasions, when the number of design points in the design set was small, a relatively poor approximation was obtained. Obviously where one is picking only a small number of points using a random number generator, there is a chance that a bad set of points can be generated and indeed on these two occasion a poor selection of points was made. In general however, when more design points are randomly selected, those points should be scattered throughout the design space and good approximations should be obtained. In conclusion, randomly selecting design points may be a viable method of design selection.
4.4 Larger Problems

Consider a problem in two variables and consider that the potential design points will be taken from a 6 x 6 grid of points. Let

\[ r = \text{total number of design points in the set of potential design points}, \]
\[ c = \text{number of design points in the selected subset of design points}, \]
\[ nc = \text{the number of different combinations of designs in the subset}. \]

For the problem at hand, \( r=36 \). Subset sizes of \( c=15, 20, 25, \) and 30 are to be considered. The number of possible combinations of design points in the subset, \( nc \), is given by

\[ nc = \frac{r!}{(r-c)! c!} \tag{70} \]

Table 4.1 summarizes the number of combinations for this study.

<table>
<thead>
<tr>
<th>Total number of design points</th>
<th>Number of point in subset</th>
<th>Number of combinations</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>15</td>
<td>5,567,902,560</td>
</tr>
<tr>
<td>36</td>
<td>20</td>
<td>7,307,872,110</td>
</tr>
<tr>
<td>36</td>
<td>25</td>
<td>600,805,296</td>
</tr>
<tr>
<td>36</td>
<td>30</td>
<td>1,947,792</td>
</tr>
</tbody>
</table>
One can see that for even small problems, it is infeasible to examine all possible combinations of subsets of size $N$ from a given set of design points. Welch [23], instead of evaluating all possible $N$-point designs, developed a "branch and bound" algorithm which guarantees global D-optimal designs but which does not generate and evaluate all possible designs. However, even here the computing costs are high. Fedorov [24] developed another technique which neglects the integer character of the components of the design set and obtains a discrete design which is rounded off to an exact design. Reference [22] reports that these designs are considered only approximate. The most popular algorithm seems to be DETMAX by Mitchell [25]. Quoting reference [22], "The algorithm starts with an initial $m$-point ED (experimental design); the final goal is an optimal $N$-point ED. During each iteration step that candidate point, which results in the largest increase of $\det(M)$, is added to the design, and subsequently that point, which results in the smallest decrease of $\det(M)$, is removed from the design. The number $m$ of points in the initial design may be larger or smaller than $N$. If necessary the algorithm first adds (if $m < N$) or rejects (if $m > N$) points until the number of points in the ED is equal to $N$. In order to avoid local optima the algorithm is able to perform 'excursions', in which several points are added at one go and subsequently the number of points is reduced to $N$. If the resulting $N$-point ED has not been improved, another excursion will be made from the same initial design. If the excursion is successful the resulting ED will be used as starting ED in a further attempt to maximize $\det(M)$. The algorithm terminates when, after several excursions, no better ED
is found. The algorithm generates high quality EDs against relatively low computing costs."

An attempt is being made to obtain the algorithm DETMAX.

4.5 Optimality Criteria Based on Minimizing Uncertainty

Reference [21] considers problems where there is no experimental error. That reference uses an optimality criteria based on selecting a design which minimizes the uncertainty in the approximating function. That reference was given mixed reviews by a number of leading authorities in the field [21] (reviews follow the paper). The formulation is quite theoretical and difficult to follow. The formulation seems to have promise but requires additional theoretical development before it becomes operative.

4.6 Conclusion

There is little rational for using any of the investigated optimality criteria when building approximations of functions which contain no experimental error. However, the D-optimality criteria can conveniently be used as a heuristic in selecting design points.

Previous investigations have indicated that approximations should be over-determined. That is to say that more training pairs should be used to build an approximations than the number of associated undetermined parameters. It has been suggested that a 20-50% over-determined system might be reasonable. The program DESIGNS, described in Section 2, develops enough designs to exactly determine a quadratic approximation of a given function. The D-optimality criteria can be used as a heuristic for selecting design points to
supplement those generated by DESIGNS. The use of the D-optimality criteria to select the supplementary points would guarantee than no singular matrices would be encountered in determining the undetermined parameters associated with the polynomial approximation.
5. Significance Testing of Coefficients

5.1 Introduction

When the training pairs used to build a polynomial response surface contain experimental or numerical error, certain coefficients in the polynomial approximation may not be significant. In other words, even though one calculates a value for some coefficient, $b_i$, the experimental or numerical error may have such an effect on that coefficient that it could just as well be taken as zero as the value calculated. In situations like this, it may be advantageous to drop that term from the polynomial approximation and redevelop the response surface. Such a procedure is discussed in pages 34-38 of [3] and an automated procedure for performing such an operation was developed in [26]. Testing of significance involves the t-test which is next described.

5.2 t-test

Coefficients of the polynomial approximation are found from Equation (12). The determination of those coefficients involve the matrix $[H]$ where

$$[H] = ([Z][Z])^{-1}$$  \hspace{1cm} (71)

A number of terms must now be defined:

$$\text{mean square error} = MSE = \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N-m}$$  \hspace{1cm} (72)
standard error coefficient = se_i = \sqrt{\text{MSE } H_i} \tag{73}

\[ t_i = \frac{|b_i|}{se_i} \tag{74}\]

where

\[ N = \text{the number of design points and } \]

\[ m = \text{the number of coefficients in the polynomial approximation.} \]

In making the test of significance, \( t_i \) from Equation (74) is compared to tabulated values of \( t_a \). The value of \( t_a \) is taken from a table of "Percentage Points of the Student's t Distribution" [3]. The value taken depends on the level of significance desired. In lieu of using tabulated values, \( t_a \) is often taken as four [26]. If \( t_i \) is less than \( t_a \) (\( t_i < t_a \)), then that coefficient's importance in approximating the response is deemed to be insignificant and therefore may be eliminated from the response function.

The primary focus of this study was to examine methods of developing good response surfaces for deterministic models, i.e. for systems that contain no experimental or numerical error. Statistical testing of coefficients presupposes experimental or numerical error and thus is not relevant when approximating response which contains no error. However, the method was thought to perhaps offer a heuristic for improving the quality of a response surface even if experimental or numerical errors are not present. Thus, two examples were
examined. Results are next reported.

5.3 Example 1 -- Fox's Banana Function

Example 1 again examines Fox's Banana Function [16]. A complete second order polynomial approximation \((m=6)\) and a complete third order polynomial approximation \((m=10)\) were developed. These approximations were developed using a complete \(6^2\) factorial design \((N=36)\). A \textit{t-value}, \(t_i\), was calculated for each parameter, \(b_i\), and compared to \(t_a = 4\). Parameter that lack significance \((t_i < t_a)\) were eliminated. A new approximation was then developed using only the significant parameters. The values of \(v\) and \(v_G\) from Equations (5) and (6), respectively, were developed for the complete polynomial and for the polynomial containing only terms deemed significant. Results are shown in Figures 25 and 26. One can see in these figures that eliminating coefficients deemed insignificant had an adverse effect on the quality of the approximation over the region of interest.

5.4 Example 2

The effect of eliminating coefficients deemed insignificant was tested on the function

\[
Y=(4+x_1)^3+\sin\left[\frac{3\pi}{2}(x_1+1)\right]+2+x_2^4+\sin\left(\frac{\pi}{2}\right)+7x_2x_1
\] (75)

Again, a complete second order polynomial approximation \((m=6)\) and a complete third order polynomial approximation \((m=10)\) were developed. These approximations were developed using a complete \(6^2\) factorial design \((N=36)\). A \textit{t-value}, \(t_i\), was calculated for
each parameter, $b_i$, and compared to $t_a = 4$. Parameter that lack significance ($t_i < t_a$) were eliminated. A new approximation was then developed using only the significant parameters. The values of $v$ and $v_G$ from Equations (5) and (6), respectively, were developed for the complete polynomial and for the polynomial containing only terms deemed significant. Results are shown in Figures 27 and 28. One can see in these figures that eliminating coefficients deemed insignificant offered no improvement in the quality of the response surface.

5.5 Conclusion

The applicability of significance testing of polynomial coefficients when modeling deterministic systems was considered. Two examples were examined to see if eliminating terms of polynomial approximations which were deemed to be insignificant by the t-test would improve the quality of the response surfaces developed. Based on these two examples, it was concluded that no improvement in the predictive capability of response surfaces over regions of interest would be obtained with such a procedure. The relevance of significance testing is when modeling systems containing numerical or experimental error.
6. Applicability of the Response Surface Technique

6.1 Introduction

The following study was performed to ascertain under what circumstances could the response surface technique be used to advantage in engineering optimization application. In this regard, assume that a quadratic polynomial approximations is to be made of functions of \( n \) variables. The number of undetermined coefficients in that approximation is:

\[
\text{number of coefficients} = \frac{(n+1)(n+2)}{2}
\]  

Equation (76)

Previous studies [19] have shown that the best approximations are obtained when the approximations are over-determined. Thus, the number of functional evaluations required to make the approximation is:

\[
\text{number of functional evaluations} = \frac{\delta(n+1)(n+2)}{2}
\]  

Equation (77)

where \( \delta \) determines the degree that the approximation is over-determined.

The functional evaluations required to build the approximation are initially performed before the start of the optimization process. By using parallel processing, these functional evaluations may be less computationally expensive than evaluations made sequentially in a direct optimization procedure. The number of required evaluations of Equation (77) is then
equivalent to a reduced number of sequential evaluations thus:

\[ \text{equivalent number functional evaluations} = \frac{\delta \beta (n+1)(n+2)}{2} \]  \hspace{1cm} (78)

where \( \beta \) is a coefficient of efficiency associated with parallel processing.

An optimum solution can be attempted using the response surfaces developed instead of the original functions. However, because of the inexact nature of the approximations, a new set of response surfaces may have to be developed at the most recent approximate solution and another optimal solution attempted. This procedure may have to be repeated \( \alpha \) times to reach the optimum solution for the original problem. The total number of equivalent functional evaluations performed in reaching this optimum is:

\[ \text{total equivalent functional evaluations} = \frac{\alpha \beta \delta (n+1)(n+2)}{2} \]  \hspace{1cm} (79)

If the solutions was attempted by direct optimization techniques instead of using response surfaces, Barthelemy [27] states that a solution can be obtained in most cases using no more than \( \psi \) first derivative evaluations. If the first derivatives are obtained by finite difference formulae, an estimate of the number of functional evaluations required by a direct solution procedure is:

\[ \text{functional evaluations direct methods} = \psi (n+1) \]  \hspace{1cm} (80)
If the response surface technique is to be competitive with the direct solution technique, then from Equations (4) and (5) one must have:

$$\frac{\alpha \beta \delta (n+1)(n+2)}{2} < \gamma \psi (n+1)$$  \hspace{1cm} (81)

where \( \gamma \) is a convenience factor associated with using response surfaces. In other words, an investigator may tolerate more functional evaluations with the response surface technique than with the direct solution procedure just for the convenience of using response surfaces.

Rearranging Equation (81) gives

$$[n+1] \left[ \frac{\alpha \beta \delta (n+2)}{2 \gamma} - \psi \right] \leq 0$$  \hspace{1cm} (82)

Since \((n+1)\) is positive, one obtains

$$\frac{\alpha \beta \delta (n+2)}{2 \gamma} - \psi \leq 0$$  \hspace{1cm} (83)

or

$$n \leq \frac{2 \psi \gamma}{\alpha \beta \delta} - 2$$  \hspace{1cm} (84)
In review

\[ \alpha = \text{number sequential optimizations} \]
\[ \beta = \text{parallel processing coefficient} \]
\[ \delta = \text{overdetermined coefficient} \]
\[ \gamma = \text{convenience coefficient} \]
\[ \psi = \text{direct solution coefficient} \]  

(85)

Reasonable ranges of the parameters involved are

\[ \alpha = 1.00 - 4.00 \]
\[ \beta = 0.10 - 1.00 \]
\[ \delta = 1.25 - 1.75 \]
\[ \gamma = 1.00 - 3.00 \]
\[ \psi = 6.00 - 10.0 \]  

(86)

For an approximate upper bound on the number of design variable that could be economical used with the response surface technique take:

\[ \alpha = 1.00 \]
\[ \beta = 0.10 \]
\[ \delta = 1.25 \]
\[ \gamma = 3.00 \]
\[ \psi = 10.0 \]  

(87)

giving

\[ n \leq 498 \]  

(88)

Under the most unfavorable set of circumstances, that is:
Thus depending upon the problem, one could use the response surface technique for $n = 0$ to $n = 500$ variables. Consider the following reasonable set of parameters

$$\alpha = 3.00 \quad \beta = 0.50 \quad \delta = 1.25 \quad \gamma = 1.50 \quad \psi = 8.00$$

(91)

Thus, it is reasonable to assume that the response surface technique could be used for up to 10-15 design variables.
6.2 Conclusion

Under the most advantageous circumstances, the response surface technique applied to engineering optimization application could be used for up to 500 design variables. Under the worst set of circumstances, it is entirely inappropriate. Under normally expected circumstances, this technique might be used to advantage for 10-15 design variables.
7. Additional Examples

7.1 Introduction
The next several examples examine the effect of design selection on the quality of approximations. In each case, a second order polynomial approximation is made of a trial function. Different number of design variables are considered in each example. Thus, for each example different designs are appropriate. In the first example, there are 4 design variables. When there are fewer than 6 design variables, central composite designs are a possible appropriate choice. Other choices are the $3^k$ factorial design, the minimum point design, the augmented minimum point design, or randomly selected design. All of these designs are considered in that example. In the second and third examples, there are 15 and 20 design variables, respectively. Here, the $3^k$ factorial design and central composite designs contain too many design points to be practical. For these examples, the minimum point design, the augmented minimum point design, and the randomly selected design are appropriate and are considered.

7.2 The 35 Bar Truss with 4 Design Variables
In many response surface applications, the function to be approximated is a relatively smooth function of the design variables which can be approximated with a lower order polynomial or an artificial neural net with only a few nodes on the hidden layer. A problem of this type is shown in Figure 29. In this example, all loads shown in Figure 29 are in kips, all members of the lower chord of the truss are assumed to have area, $A_1$, and all members
of the upper chord to have area, \( A_2 \), all vertical and diagonal members to have area, \( A_3 \).

The depth of the truss is \( H \). A response surface is to be constructed for the stress in member BC in terms of the design variables, \( x_i \) thus

\[
x_i = \frac{1}{A_i}, \quad i = 1, 3
\]

\[
x_4 = 0.09375H - 0.4375
\]

The region of interest is

\[
2 \text{ in}^2 \leq A_i \leq 8 \text{ in}^2
\]

\[
6 \text{ ft} \leq H \leq 10 \text{ ft}
\]

or in terms of the design variables

\[
0.125 \leq x_i \leq 0.5
\]

A number of designs were used to develop a second order polynomial approximation for the stress in member BC. Each approximation was then used to predict stress on a 5 x 5 x 5 x 5 grid of points. The predicted stress and the actual stress on these \( \text{NG} = 625 \) grid of points were then used to develop \( v_G \) from Equation (6). The parameter \( v_G \) is a measure of the quality of the approximation over the region of interest.

The different designs examined required different numbers of functional evaluation. So as to get a measure of the quality of fit of the approximation over the region of interest which
Table 7.1 The 35 bar truss with 5 design variables, 2nd order polynomial approximation

<table>
<thead>
<tr>
<th>Description</th>
<th>m</th>
<th>α</th>
<th>T</th>
<th>F</th>
<th>v (%)</th>
<th>v_G (%)</th>
<th>E_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>3^4 factorial design</td>
<td>...</td>
<td>...</td>
<td>81</td>
<td>81</td>
<td>3.34</td>
<td>2.41</td>
<td>1.00</td>
</tr>
<tr>
<td>single center point rotatable central composite design</td>
<td>1</td>
<td>2.000</td>
<td>25</td>
<td>25</td>
<td>0.66</td>
<td>2.67</td>
<td>0.34</td>
</tr>
<tr>
<td>multiple center point rotatable uniform precision central composite design</td>
<td>7</td>
<td>2.000</td>
<td>31</td>
<td>25</td>
<td>0.59</td>
<td>2.67</td>
<td>0.34</td>
</tr>
<tr>
<td>single center point orthogonal central composite design</td>
<td>1</td>
<td>1.414</td>
<td>25</td>
<td>25</td>
<td>1.47</td>
<td>2.37</td>
<td>0.30</td>
</tr>
<tr>
<td>multiple center point rotatable orthogonal central composite design</td>
<td>12</td>
<td>2.000</td>
<td>36</td>
<td>25</td>
<td>0.55</td>
<td>2.67</td>
<td>0.34</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS</td>
<td>...</td>
<td>...</td>
<td>15</td>
<td>15</td>
<td>0.00</td>
<td>3.99</td>
<td>0.31</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS augmented by 3 randomly selected design points</td>
<td>...</td>
<td>...</td>
<td>18</td>
<td>18</td>
<td>0.40</td>
<td>3.86</td>
<td>0.36</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS augmented by 6 randomly selected design points</td>
<td>...</td>
<td>...</td>
<td>21</td>
<td>21</td>
<td>0.38</td>
<td>3.91</td>
<td>0.42</td>
</tr>
<tr>
<td>minimum point design from program DESIGNS augmented by 9 selected design points</td>
<td>...</td>
<td>...</td>
<td>24</td>
<td>24</td>
<td>0.41</td>
<td>3.77</td>
<td>0.46</td>
</tr>
<tr>
<td>randomly selected design</td>
<td>...</td>
<td>...</td>
<td>25</td>
<td>25</td>
<td>0.00</td>
<td>824.2</td>
<td>105</td>
</tr>
</tbody>
</table>

m = number of design points at the center of the design space  
T = the total number of design points  
F = the number of functional evaluations required  
α = parameter which defines location of certain design points
takes into account the number of functional evaluations performed, the efficiency, $E_j$, from
Equation (64) was developed for each design. Table 7.1 reports for each design considered,
the efficiency, $E_j$, as well as other relevant information.

One can see in Table 7.1 that all the designs considered, except the randomly selected
design, gave a good approximation over the region of interest. Randomly selected designs,
which often work well, can sometimes suffer from the problem that the coefficient matrix
used to solve for the approximation's associated parameters is poorly conditioned or that
the design points selected are not well scattered throughout the design space. In either case,
they can yield a poor approximation over the region of interest as in this example.

The $3^4$ factorial design well approximated the trial function. However, because it uses so
many design points its efficiency measure is poor and thus is not a design of choice. The
single center point orthogonal central composite design and the minimum point design from
program DESIGNS performed the best, based on their efficiency. However, excluding the
randomly selected design and the $3^4$ factorial design, all of the designs considered gave a low
value of $v_G$ and had approximately the same value of efficiency.

Under normal circumstances, information is not available to calculate $v_G$ and one must use
the parameter $v$ as a measure of the quality of fit over the region of interest. However, the
parameter $v$ is only a measure of quality of fit over the region of interest if the
approximation is over-determined. Thus, under normal circumstances one would not want
to use the minimum point design. This example indicates, that for problems of the size of this example, that any of the central composite designs or the augmented minimum point designs would be appropriate.

7.2 The 35 bar truss with 15 design variables

This example again considers the 35 bar truss of Figure 29. In this example, H is 10 ft., the areas of the 14 bars of the top and bottom chords are \( A_i \), \( i = 1,14 \), and the area of the vertical and diagonal members is \( A_{15} \). The design variables of the problem are taken as

\[
x_i = \frac{1}{A_i}, \quad i = 1,15
\]

(96)

The region of interest is

\[
2 \text{ in}^2 \leq A_i \leq 8 \text{ in}^2
\]

(97)

or in terms of the design variables

\[
.125 \leq x_i \leq .5
\]

(98)

Response surfaces were developed for the stress in member BC using a 2nd order polynomial approximation. The approximation were developed using various designs. To test the quality of the approximations over the region of interest, the function and the approximations were evaluated at \( N_G = 500 \) randomly selected test points over the region of interest. That information was then used to calculate \( v_G \) from Equation (6). The random
number generator used to develop design points uses, in generating its numbers, an initial seed parameter, IFLAG. A different value of IFLAG was used to generate the 500 test points than was used to generate random points in the randomly selected designs or in the augmented minimum point designs. Thus, the test set of points does not duplicate any of the design points in the designs considered. Results of this investigation are reported in Table 7.2.

One will notice in Table 7.2 that only minimum point designs, augmented minimum point designs, and randomly selected designs are considered. A $3^{15}$ factorial design contains over 14 million design points. Thus, the use of the $3^{15}$ factorial design is out of the question. For a problem in $k$ design variables, the central composite design uses a $2^k$ factorial design augmented by $2k+1$ additional design points. Thus, such a single center point central composite design for this problem contains 32,799 design points. Here again, such a design is impractical. One can develop a central composite design by augmenting only a fraction of the $2^k$ factorial design. For this problem, a single center point central composite design using only a 1/4 fraction of the $2^{15}$ factorial design would contain 8,223 design points which is still an impractical design. Thus, for problems of the size of this example, only the minimum point designs, augmented minimum point designs, and randomly selected designs are of reasonable size.

We can see in Table 7.2 that all of the designs with the exception of the "randomly selected-exactly determined design" did a good job of approximating truss behavior. A singular
matrix was encountered in Equation (10) for the randomly selected--exactly determined design. With completely randomly selected designs, there is always the possibility of having a poorly conditioned coefficient matrix \([Z]\) in Equation (10) and indeed this occurred in this problem. However, there was no problem with matrix conditioning using randomly selected over-determined designs.

Table 7.2 The 35 bar truss with 15 design variables, 2nd order polynomial approximation

<table>
<thead>
<tr>
<th>Description</th>
<th>(F)</th>
<th>(v%)</th>
<th>(v_G%)</th>
<th>(E_j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimum point design from program DESIGN-exactly determined</td>
<td>136</td>
<td>0</td>
<td>1.263</td>
<td>1.0</td>
</tr>
<tr>
<td>augmented minimum point design--20% over-determined</td>
<td>163</td>
<td>0.083</td>
<td>0.294</td>
<td>0.28</td>
</tr>
<tr>
<td>augmented minimum point design--40% over-determined</td>
<td>190</td>
<td>0.087</td>
<td>0.060</td>
<td>0.07</td>
</tr>
<tr>
<td>random selection--exactly determined</td>
<td>136</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>random selection--20% over-deteremined</td>
<td>163</td>
<td>0.003</td>
<td>0.029</td>
<td>0.03</td>
</tr>
<tr>
<td>random selection--40% over-determined</td>
<td>190</td>
<td>0.003</td>
<td>0.010</td>
<td>0.01</td>
</tr>
</tbody>
</table>

* singular coefficient matrix

The efficiency parameter, \(E_p\), is calculated in Table 7.2 but it is rather a meaningless parameter for this problem because all the designs so well fit the exact function. In real life
situations, one usually does not have available information for calculating $v_G$. Thus, the parameter $v$ or like term must be used as a measure of the quality of the approximation. The parameter $v$ is not a meaningful measure of the quality of fit over a region of interest unless the system is over-determined. Thus for this example, the design of choice would be either the 20% over-determined minimum point design or the 20% over-determined randomly selected design.

### 7.3 Analytical function--20 design variables

This example considers a problem with even more design variables. The function tested is:

$$y = 1 + \sum_{i=1}^{20} x_i + \sum_{i=1}^{20} \sum_{j=i+1}^{20} x_i x_j + \sum_{i=1}^{20} \sum_{j=1}^{20} x_i^2 x_j$$

A second order polynomial function was used to build the response surface approximating this function. The polynomial approximating function had 231 undetermined coefficients. Because of the large size of this problem, factorial designs and central composite designs are not appropriate. A minimum point design, augmented minimum point designs, and randomly selected designs were considered. Values of the test function and approximate function were evaluated at $NG = 1000$ randomly selected points and the parameter $v_G$ was developed using this information. The measure of efficiency of the designs examined along with other relevant information is given in Table 7.3.
Table 7.3 Analytical function with 20 design variables, 2nd order polynomial approximation

<table>
<thead>
<tr>
<th>Description</th>
<th>F</th>
<th>v %</th>
<th>(v_G) %</th>
<th>(E_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimum point design from program DESIGN-</td>
<td>231</td>
<td>0</td>
<td>88.93</td>
<td>1.0</td>
</tr>
<tr>
<td>exactly determined</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>augmented minimum point design--20% over-</td>
<td>277</td>
<td>5.83</td>
<td>49.82</td>
<td>0.67</td>
</tr>
<tr>
<td>determined</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>augmented minimum point design--40% over-</td>
<td>323</td>
<td>9.58</td>
<td>18.03</td>
<td>0.28</td>
</tr>
<tr>
<td>determined</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>random selection--exactly determined</td>
<td>231</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>random selection--20% over-determined</td>
<td>277</td>
<td>0.61</td>
<td>7.21</td>
<td>0.10</td>
</tr>
<tr>
<td>random selection--40% over-determined</td>
<td>323</td>
<td>0.46</td>
<td>1.20</td>
<td>0.02</td>
</tr>
</tbody>
</table>

* poorly conditioned coefficient matrix

Just as in Example 7.2, a exactly determined randomly selected design gave a poorly conditioned coefficient matrix. These examples indicate that randomly selected exactly determined designs should be avoided. The 40% over-determined randomly selected design did an excellent job of modeling the test function and was the most efficient design considered. It seems that on problems with a large number of design variables that randomly selected over-determined designs should be expected to work well.
7.4 Conclusion

The examples of this section have shown that design selection depends on the number of design variables. If the number of design variables is less than 6, appropriate designs are:

1. augmented minimum point designs
2. central composite designs
3. over-determined randomly selected designs.

When there are more than 6 design variables, the central composite designs contain too many design point for consideration. For more than 6 design variables, appropriate designs are then

1. augmented minimum point designs
2. over-determined randomly selected designs.

The example examined indicate that in all cases, over-determined designs should be used. They the most efficient designs. Also, when a design is over-determined the coefficient $v$ can be used as a measure of the quality of the approximation over a region of interest. Being able to use $v$ as a measure of the quality of fit over the region of interest is very important because, in general, information is not available to determined the parameter $v_G$. 
8. Augmented Minimum Point Designs

8.1 Introduction

Design selection in the literature concentrates on linear or quadratic response surfaces. This study has also concentrated on quadratic approximations for several reasons:

1. linear approximations, in most instances, will be inadequate to model functions of interest,

2. for many problems, a 2nd order approximation will be adequate to model response especially if the region of interest is limited,

3. there is a scarcity of literature which address design selection for cubic or higher order polynomial approximations, and

4. in optimization process using response surfaces, for moderate size problems, it is more computationally efficient to perform a sequence of quadratic approximations than one cubic or higher order approximation. This fact is next discussed.

The number of terms in a second order polynomial in n design variables is

\[
\text{number terms quadratic} = (n+1) + \frac{n(n+1)}{2}
\]  \hspace{1cm} (100)

The number of terms in a 3rd order polynomial in n design variables is
Table 8.1 gives, for various number of design variables, the number of terms in a 2nd order and 3rd order polynomial and their ratio.

Table 8.1 Number of terms in a 2nd and 3rd order polynomial and their ratio

<table>
<thead>
<tr>
<th>number of design variables, n</th>
<th>number of terms in quadratic</th>
<th>number of terms in cubic</th>
<th>cubic/quadratic</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>10</td>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>28</td>
<td>84</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>55</td>
<td>220</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>91</td>
<td>455</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>136</td>
<td>816</td>
<td>6</td>
</tr>
</tbody>
</table>

One can see that for problems with more than 6 design variables, it will probably be more computationally efficient in an optimization algorithm to utilize a sequence of quadratic response surfaces than one 3rd or higher order response surface. When there are 6 or fewer design variables, 3rd or 4th order response surfaces may be used to advantage.

In this report, the term "minimum point design" refers to a design that has just enough design points to allow the determination of coefficients of an approximating polynomial. The term "augmented minimum point design" is a minimum point design which contains
additional design points. Thus, augmented minimum point designs are over-determined designs. The studies that have been performed in this report indicate that augmented minimum point designs are competitive with, if not better than, central composite designs for developing a 2nd order response surface. A program DESIGNS [20] was developed for generating augmented minimum point designs for developing a 2nd order response surface. That program is described in Section 8.2.

When there are 6 or fewer design variables, it may be computationally beneficial to use a 3rd order or 4th order response surface. Thus, the program DESIGN4 [28] was developed to generate augmented minimum point designs for a 4th order response surface. The program DESIGN4 is discussed in Section 8.3. The program can also be used to develop a 3rd order response surface. The 3rd order minimum point design is a subset of the 4th order minimum point design. Thus the 4th order minimum point design will give an over-determined 3rd order approximation. Additional randomly selected design points can be added to the 4th order minimum point design to give the desired degree that the 3rd order approximation is to be over-determined.

8.2 Augmented Minimum Point Designs for 2nd Order Approximations

The basic building block for program DESIGNS is the star pattern of design points. Figure 4 shows the star pattern for 3 design variables. This pattern of design points allows one to determine those coefficients of a 2nd order polynomial approximation associated with the
To be able to determine the coefficients associated with the terms

$$1, x_i, x_i^2, \quad i=1,n$$

one must supplement the star pattern with one additional design point in the $x_i, x_j$ planes. Figure 30 shows the additional design point in the $x_i, x_j$ plane. Figure 6 shows the total minimum point design for 3 design variables.

Studies of this report indicate that designs should be over-determined. Having a design that is 20-50% over-determined is a good compromise between keeping down the number of design points while still getting a good approximation. The program DESIGNS augments the minimum point design with a user selected number of random design points.

### 8.2.1 Specifics of program DESIGNS

A listing of the FORTRAN program DESIGNS is found in Appendix 1 and a copy of that program is found in file "designs.f" on the floppy disk accompanying this report. The program should be compiled with a F77 compiler with the compiled program called "design". To run the program just enter "design" from the keyboard. The program prompts the user for
1. the number of design variables,
2. the number of designs points to augment the minimum point design, and
3. a seed parameter, IFLAG, which is used to generate the random numbers (IFLAG can
   be entered as any positive integer).

The program then generates a design in local coordinates with the maximum range on each
design variable of -1 to +1. The program then
4. asks the user to enter an integer which specifies whether design point coordinates are
to be also generated in global coordinates. If they are to be calculated in global
coordinates, the program then
5. prompts the user to enter the range of design variables in global coordinates.

Results with commentary are written to file "design.res". Design points without commentary
are written to file "design.run".

8.3 Augmented Minimum Point Design for 3rd and 4th Order Approximation

A $3^k$ factorial design is used as the building block of this minimum point design. The $3^k$
factorial design provides information for calculating the coefficients associated with the
terms

\[ 1, x_p x_i, x_i^2, x_i x_p, x_i^2 x_j, j \neq i \quad (104) \]

Additional points are then added at -1 and 1 (in local coordinates) along the $x_i$ axis. These
points together with the $3^k$ factorial design point give the star pattern which can be seen in Figure 31. With this arrangement of points, there are 5 design points along the $x_i$ axis which provides information for calculating the coefficient associated with the terms

$$x_i^4$$  \hspace{1cm} (105)

Additional design points are then placed in each $x_i, x_j$ plane which provides information for calculating the coefficient associated with the terms

$$x_i^3 x_j$$  \hspace{1cm} (106)

These points are also shown in Figure 31.

8.3.1 Specifics of program DESIGN4

A listing of the FORTRAN program DESIGN4 is found in Appendix 2 and a copy of that program is found in file "design4.f" on the floppy disk accompanying this report. The program should be compiled with a F77 compiler with the compiled program called "design4". To run the program just enter "design4" from the keyboard. The program
prompts the user for needed information. Prompts and response are similar to those for the program DESIGNS.

8.4 Conclusion

A minimum point design is a design that has just enough design points to allow the determination of the coefficients of an approximating polynomial. An augmented minimum point design is a minimum point design which contains additional design points. Augmented minimum point designs are competitive with, if not better than, central composite designs for developing a 2nd order response surface. Minimum point designs should be augmented with enough points that the approximation is 20-50% over-determined. A program DESIGNS was developed for generating augmented minimum point designs for developing a 2nd order response surface.

When there are more than 6 design variables, 3rd or higher order approximations require so many design points that it is computationally better to perform a sequence of 2nd order approximations in an optimization process than one higher order approximation. When there are 6 or fewer design variables, a 2nd order approximation may often be satisfactory. However, for those cases where it is desirable to use a higher order approximation, program DESIGN4 was developed. That program generates designs which can be used to develop 3rd or 4th order approximations.
9. Solution Algorithm

9.1 Introduction

In this investigation, the program NEWPSI was used to perform the studies involving polynomial approximations. That program can investigate under-determined, exactly-determined, or over-determined approximations of various orders. The version submitted with this report can handle up to 15 design variables as programmed. The order of polynomial it can handle is as follows:

1. one design variable, up to a 20th order polynomial
2. two design variables, up to a 5th order polynomial
3. for 2-15 design variables, a second order polynomials.

One can use up to 250 designs to train the approximation. In calculating \( v_G \), it can handle up to 2000 grid points.

The program solves for the undetermined parameters associated with the approximation. It then evaluates the approximate function at the design points and calculates the error parameter, \( v \). It then reads in the design points and function value on the test grid. The approximate function is evaluated at the grid points and the error parameter, \( v_G \), is then evaluated.

9.2 Program Specifics

A listing of the FORTRAN program NEWPSI is found in Appendix 3 and a copy of that
program is found in file "newpsi.f" on the floppy disk accompanying this report. The program should be compiled with a F77 compiler and the compiled program called "newpsi". To run the program just enter "newpsi" from the keyboard. Data is read from the file "newpsi.dat". Data can be in free format. The program asks for the following data:

1. a value of the print code, ip; (If ip=4, great quantities of output are generated for program checkout. Normally the program is run with ip=0 for normal output).

2. the number of design variable, nd;

3. the order of the polynomial being considered, np;

4. the number of design points in the design, m;

5. the design and function value at the design points, x(i,j), y(i);

6. the number of design points on the grid, ng; and

7. the design and function value at the grid points, xx(i,j), yy(j).

Output is written to the screen and to file "newpsi.res".
10. Conclusion

For a given order of approximation of a function, \( f \), the quality of the approximation is affected by

a. the number of levels of the design variables,

b. the location of the design points, and

c. the degree which the approximation is over-determined.

For an \( n \)th order approximation,

1. there must be \( n+1 \) levels of the design variables;

2. the design points must be located so that information is available for calculating all of the \( n \)th derivatives of \( f \);

3. the approximation should be, at least, 20-50\% over-determined.

For example, for a 2nd order approximation in 3 design variables, there must be at least 3 levels of the design variables, design points must be located so that information is available for calculating

\[
\frac{\partial f}{\partial x_i}, \frac{\partial^2 f}{\partial x_i \partial x_j}, \quad i=1,3; \ j=1,3
\]  

(107)

A complete 2nd order polynomial approximation contains 10 undetermined coefficients. Thus, at least 10 design points are required to provide information for calculating these
coefficients. To have the approximation 30% over-determined, one would want to use 13
design points.

For second order approximations, when there are fewer than 6 design variables, central
composite designs meet requirements 1-3. However, for 6 or more design variables, these
designs contain too many design points. A minimum point design is one which contains just
enough design points, meeting the derivative requirements of item 1 and 2 above, to exactly-
determine the approximation. An augmented minimum point design is a minimum point
design supplemented with additional design points. The program DESIGNS was developed
to yield augmented minimum point designs for 2nd order approximations. The quality of
approximations developed using designs from program DESIGNS was comparable to, if not
better than, other standard designs such as the central composite designs.

For more than 6 design variables, 3rd and 4th order approximations require so many design
points to determine the coefficients in those approximations that it is more computationally
efficient to develop a number of 2nd order approximations than one approximation of 3rd
or higher order. For 6 or fewer design points, 2nd order approximations may be quite
adequate. However, for those cases where one wishes to use a 3rd or 4th order
approximation, the program DESIGN4 was develop. That program generates an augmented
minimum point design for developing a 4th order approximation.

Previous studies have shown that the quality of approximations using neural networks is
comparable to those using polynomial approximations when the number of undetermined parameters associated with the approximations is the same. Thus, neural networks trained with designs from DESIGNS or DESIGN4 should offer approximations of comparable quality to those obtained using polynomial approximations with the same number of associated undetermined parameters.
11. References


13. Hornik, Kurt, et. al., "Multilayer Feedforward Networks are Universal


Figure 1. Artificial neural net
Figure 2. Complete design
Figure 3. Deficient design
Figure 5. Star pattern of design points—12 design points
Figure 7. Central composite design for $k=3$
Figure 8. Fox's banana function
One Dimensional Example

Figure 9. One dimensional example
Parameter vG
First Order Polynomial Approximation

Figure 10. D, A, E, and G optimality, first order approximation
Parameter vG
Second Order Polynomial Approximation

Figure 11. D, A, E, and G optimality, second order approximation
Parameter $vG$
Third Order Polynomial Approximation

Figure 12. D, A, E, and G optimality, third order approximation
Parameter $v_G$
Fourth Order Polynomial Approximation

![Graph showing parameter $v_G$ for different optimality criteria.](image)

Figure 13. D, A, E, and G optimality, fourth order approximation
Parameter $v_G$
First Order Polynomial Approximation

Figure 14. $S$ and $Q$ optimality, first order approximation
Parameter vG
Second Order Polynomial Approximation

Figure 15. S and Q optimality, second order approximation
Parameter vG
Third Order Polynomial Approximation

Figure 16. S and Q optimality, third order approximation
Parameter $v_G$

Fourth Order Polynomial Approximation

Figure 17. S and Q optimality, fourth order approximation
Y and its Approximation
Q optimality, 11 points out of 13

Figure 18. Q optimality, 11 out of 13 points selected
Y and its Approximation
Q optimality, 7 out of 13 points

Figure 19. Q optimality, 7 out of 13 points selected
Y and its Approximation

Q optimality, 5 out of 13 points

Figure 20. Q optimality, 5 out of 13 points selected
Parameter vG
First Order Polynomial Approximation

Figure 21. Random points, first order approximation
Parameter $v_G$
Second Order Polynomial Approximation

Figure 22. Random points, second order approximation
Parameter \( v_G \)
Third Order Polynomial Approximation

Figure 23. Random points, third order approximation
Parameter vG
Fourth Order Polynomial Approximation

Number of designs in subset

Figure 24. Random points, fourth order approximation
MEASUREMENTS OF QUALITY OF FIT BEFORE AND AFTER t-test PERFORMED ON

\[ y = 10x_1^4 - 20x_2x_1^2 + 10x_2^2x_1^2 + x_1^2 - 2x_1 + 5 \]

"Fox's Banana Function"

SECOND ORDER APPROXIMATION

\[ Y = b_0 + b_1x_1 + b_2x_2 + b_3x_1^2 + b_4x_1x_2 + b_5x_2^2 \]

Before t-test

\[ v: \quad 26.8 \]
\[ V_c: \quad 102.11 \]

Solution of Coefficients

\[ b = \begin{pmatrix} 121.2 \\ -836.3 \\ 66.7 \\ 393.9 \\ -100 \\ 10 \end{pmatrix} \]

After t-test

\[ v: \quad 41.8 \]
\[ V_c: \quad 175.82 \]

Solution of Coefficients

\[ b = \begin{pmatrix} 0 \\ -814.0 \\ 0 \\ 352.6 \\ 0 \end{pmatrix} \]

Figure 25. Significance testing, Example 1, 2nd order approximation
MEASUREMENTS OF QUALITY OF FIT
BEFORE AND AFTER t-test
PERFORMED ON
\[ y = 10x_1^4 - 20x_2x_1^3 + 10x_2^2x_1^2 + x_1^2 - 2x_1 + 5 \]
"Fox's Banana Function"

THIRD ORDER APPROXIMATION

\[ y = b_0 + b_1x_1 + b_2x_2 + b_3x_1^2 + b_4x_1x_2 + b_5x_2^2 + b_6x_1^3 + b_7x_1^2x_2 + b_8x_1x_2^2 + b_9x_2^3 \]

Before t-test

\[ v : 2.9 \]
\[ V_c : 53.71 \]

Solution of Coefficients

\[ b = \begin{bmatrix} -12.1 \\ 283.7 \\ 0 \\ -306.1 \\ 0 \\ 10 \\ 100 \\ -20 \\ 0 \\ 0 \end{bmatrix} \]

After t-test

\[ v : 6.4 \]
\[ V_c : 112.38 \]

Solution of Coefficients

\[ b = \begin{bmatrix} 0 \\ 385.0 \\ 0 \\ -349.3 \\ 0 \\ 103.8 \\ -17.2 \\ 0 \\ 0 \end{bmatrix} \]

Figure 26. Significance testing, Example 1, 3rd order approximation
MEASUREMENTS OF QUALITY OF FIT
BEFORE AND AFTER t-test
PERFORMED ON

\[ Y = (4 + x_1)^3 + \sin\left(\frac{3\pi}{2} \cdot (x_1 + 1)\right) + 2 + x_2^4 + \sin\left(\frac{\pi}{2}\right) + 7x_2x_1 \]

SECOND ORDER APPROXIMATION

\[ Y = b_0 + b_1x_1 + b_2x_2 + b_3x_1^2 + b_4x_1x_2 + b_5x_2^2 \]

Before t-test

\[ v: \ 6.2 \]
\[ V_c: \ 90.02 \]

Solution of Coefficients

\[
\begin{bmatrix}
97.6 \\
35.0 \\
-108.4 \\
19.4 \\
7 \\
44.3
\end{bmatrix}
\]

After t-test

\[ v: \ 8.6 \]
\[ V_c: \ 123.67 \]

Solution of Coefficients

\[
\begin{bmatrix}
96.4 \\
0 \\
-90.9 \\
29.0 \\
0 \\
44.3
\end{bmatrix}
\]

Figure 27. Significance testing, Example 2, 2nd order approximation
MEASUREMENTS OF QUALITY OF FIT
BEFORE AND AFTER t-test
PERFORMED ON

\[ Y = (4 + x_1)^3 + \sin \left( \frac{3\pi}{2} \cdot (x_1 + 1) \right) + 2 + x_1^4 + \sin \left( \frac{\pi}{2} \right) + 7x_2x_1 \]

THIRD ORDER APPROXIMATION

\[ Y = b_0 + b_1x_1 + b_2x_2 + b_3x_1^2 + b_4x_1x_2 + b_5x_2^2 + b_6x_1^3 + b_7x_1^2x_2 + b_8x_1x_2^2 + b_9x_2^3 \]

Before t-test After t-test

| \( v \) | 0.7 | 0.7 |
| \( V_c \) | 27.87 | 29.92 |

Solution of Coefficients

Before t-test After t-test

\[ b = \begin{bmatrix} 64.1 \\ 50.7 \\ 28.6 \\ 10.8 \\ 7 \\ -30.7 \\ 1.2 \\ 0 \\ 0 \\ 10 \end{bmatrix} \quad b = \begin{bmatrix} 64.1 \\ 50.8 \\ 28.6 \\ 10.8 \\ 7 \\ -30.7 \\ 0 \\ 0 \\ 0 \\ 10 \end{bmatrix} \]

Figure 28. Significance testing, Example 2, 3rd order approximation
Figure 29. The 35 bar truss
Figure 30. Additional points to complete a second order design.
added to find coefficients of terms of Eq. (106)

added to complete star pattern

Figure 31. Additional points to complete a fourth order design
PROGRAM DESIGNS

PROGRAM TO GENERATE DESIGNS FOR 2ND ORDER POLYNOMIAL
PROGRAM DIMENSIONED FOR UP TO 20 VARIABLES
RESULTS TO SCREEN AND TO FILE designs.res
DESIGN IN GLOBAL COORDINATES TO FILE designs.run

DEFINITIONS
N = NUMBER OF DESIGN VARIABLES
M = NUMBER OF RANDOM DESIGNS POINTS

DIMENSION X(2000,20)
DIMENSION XBB(20),XBE(20),A(20),B(20)

1 FORMAT(I5,6FI0.6)
2 FORMAT(' PROGRAM GENERATES DESIGNS FOR FITTING 2ND ORDER',
X' POLYNOMIAL')
3 FORMAT(' ENTER NUMBER OF DESIGN VARIABLES')
4 FORMAT(' NUMBER OF DESIGN VARIABLES = N =', I3)
11 FORMAT(6F10.6)

OPEN(UNIT=7,FILE='designs.res')
OPEN(UNIT=8,FILE='designs.run')
WRITE(6,2)
WRITE(6,3)
READ(5,*)N
WRITE(6,4)N

SET UP TERMS
NP1=N+1
NM1=N-1
M=(N*N+3*N+2)/2
MP1=M+1

ZERO DESIGN MATRIX
DO100I=1,M
DO100J=1,N
100 X(I,J)=0.
II=0

GENERATE THE FIRST N+1 POINTS FOR FITTING A LINEAR FUNCTION
THE FIRST POINT IS WHEN ALL X'S ZERO, ALREADY DONE
GENERATE NEXT N POINTS
DO101I=1,N
II=I+1
101 X(II,I)=1.

GENERATE NEXT N POINTS
THE 2N+1 POINTS THUS GENERATED WILL ALLOW ADDING SQUARED TERMS
DO102I=1,N
II=I+N+1
102 X(II,I)=-1.

GENERATE NEXT N(N-1)/2 POINTS
THE (N*N+3*N+2)/2 POINTS THUS GENERATED WILL ALLOW ADDING CROSS
PRODUCT TERMS. WE WILL THEN HAVE COMPLETE 2ND ORDER POLYNOMIAL
APPROXIMATION
ILAST=2*N+1
IDO=N-1
J=1
JJ=2
103 CONTINUE
DO104I=1,IDO
II=I+ILAST
X(II,J)=1.
X(II,JJ)=1.
JJ=JJ+1
104 CONTINUE
ILAST=ILAST+IDO
IDO=IDO-I
J=J+1
JJ=J+1
IF(J.LE.NM1)GOTO103

IF WE GOT HERE WE HAVE DEVELOPED THE MINIMUM POINT DESIGN

WRITE(6,*)' WE HAVE GENERATED ',II,' POINTS IN THE MIN PT DESIGN'
WRITE(7,*)' WE HAVE GENERATED ',II,' POINTS IN THE MIN PT DESIGN'
WRITE(6,*)' DESIGN POINTS WRITTEN TO FILE designs.res'

DEVELOP DESIGN POINTS TO AUGMENT THE MINIMUM POINT DESIGN
READ IN THE NUMBER OF RANDOM DESIGN POINTS TO BE DEVELOPED
WRITE(6,*)' ENTER THE NUMBER OF RANDOM GENERATED DESIGN PTS',
X' DESIRED=M'
READ(5,*)M
WRITE(6,*)' NUMBER OF RANDOM DESIGN POINTS M=',M
WRITE(7,*)' NUMBER OF RANDOM DESIGN POINTS M=',M
WRITE(6,*)' IFLAG IS ANY POSITIVE INTEGER USED TO START RANDOM',
X' PROCESS'
WRITE(6,*)' ENTER IFLAG'
READ(5,*)IFLAG
WRITE(6,*)' IFLAG=',IFLAG
WRITE(7,*)' IFLAG=',IFLAG
DO850I=I,M
II=II+1
DO851J=I,N
IFLAG=IFLAG+I
XDUM=RAND(IFLAG)
X(II,J)=2.*XDUM-I.
851 CONTINUE
850 CONTINUE

IF WE GOT HERE WE HAVE FINISHED GENERATING THE RANDOM DESIGN PTS
WRITE(6,*)' RANDOM DESIGN POINTS WRITTEN TO FILE designs.res'

PRINT OUT THE MINIMUM POINT MATRIX IN LOCAL COORDINATES

WRITE(7,*)' DESIGN MATRIX IN LOCAL COORDINATES'
ITOTAL=II
DO700I=1,ITOTAL
WRITE(7,1)I,(X(I,J),J=I,N)
700 CONTINUE

SEE IF WE ARE TO GENERATE DESIGNS IN GLOBAL COORDINATES

WRITE(6,*)' ITEST=1 IF DESIGN POINTS ARE TO BE IN GLOBAL',
X' COORDINATES'
WRITE(6,*)' OTHERWISE, ITEST=0'

WRITE(6,*)' ENTER ITEST'
READ(5,*)ITEST
IF(ITEST.NE.1)GOTO860
-C IF WE GOT HERE WE ARE TO GENERATE DESIGNS IN GLOBAL COORDINATES
WRITE(6,*)' ENTER LOWER AND UPPER RANGE ON EACH DESIGN VARIABLE'
WRITE(6,*)' i.e. ENTER XBB(I) TO XBE(I)'
DO861I=1,N
READ(5,*)XBB(I),XBE(I)
WRITE(6,*)' I,XBB(I),XBE(I)='',I,XBB(I),XBE(I)
WRITE(7,*)' I,XBB(I),XBE(I)='',I,XBB(I),XBE(I)
- 861 CONTINUE
GOTO862
860 CONTINUE
-C
-C IF WE GOT HERE LOWER BOUND VARIABLE IN GLOBAL COORDINATES IS -1
-C IF WE GOT HERE UPPER BOUND VARIABLE IN GLOBAL COORDINATES IS 1
DO863I=1,N
XBB(I)=-1.
XBE(I)=1.
CONTINUE
- 862 CONTINUE
WRITE(7,*)' I,XBB(I),XBE(I),A(I),B(I)'
DO1301I=1,N
A(I)=(XBE(I)-XBB(I))/2.
B(I)=(XBE(I)+XBB(I))/2.
WRITE(7,*)' I,XBB(I),XBE(I),A(I),B(I)
CONTINUE
DO1202I=1,ITOTAL
DO1202J=I,N
X(I,J)=A(J)*X(I,J)+B(J)
WRITE(6,*)' DESIGN IN GLOBAL COORDINATES WRITEN TO designs.res'
WRITE(6,*)' DESIGN IN GLOBAL COORDINATES WRITEN TO designs.run'
WRITE(7,*)' DESIGN IN GLOBAL COORDINATES'
WRITE(8,*)ITOTAL
DO970I=1,ITOTAL
WRITE(7,1)I, (X(I,J),J=I,N)
WRITE(8,11) (X(I,J),J=I,N)
- 970 CONTINUE
STOP
END
Program DESIGN4

Program to generate designs for 4th order polynomial
Program dimensioned for up to 6 variables
Results to screen and to file design4.res
Design in global coordinates to file design4.run

Definitions
N = number of design variables
M = number of random designs points

Dimension x(2000, 6)
Dimension xbb(10), xbe(10), a(10), b(10)

1 Format(I5, 6F10.6)
2 Format(‘ program generates designs for fitting 4th order’,
X’ polynomial’)
3 Format(‘ enter number of design variables’)
4 Format(‘ number of design variables = n = ‘, i3)
11 Format(6F10.6)
Open(unit=7, file=’design4.res’)
Open(unit=8, file=’design4.run’)
Write(6, 2)
Write(6, 3)
Read(5, *) n
Write(6, 4) n
If(n . eq. 6) goto 601
If(n . eq. 5) goto 501
If(n . eq. 4) goto 401
If(n . eq. 3) goto 301
If(n . eq. 2) goto 201
If(n . eq. 1) goto 101
Write(6, *) ‘ program can not do more than 6 design variables’
Write(7, *) ‘ program can not do more than 6 design variables’
Stop

Develop 3 factorial design to get 4 design variable product terms

101 Continue
ii = 0
Do100i1 = 1, 101, 50
ii = ii + 1
x(ii, 1) = float(i1 - 51) / 100.
100 Continue
goto 701

201 Continue
ii = 0
Do200i1 = 1, 101, 50
Do200i2 = 1, 101, 50
ii = ii + 1
x(ii, 1) = float(i1 - 51) / 100.
x(ii, 2) = float(i2 - 51) / 100.
200 Continue
goto 701

301 Continue
ii = 0
Do300i1 = 1, 101, 50
Do300i2 = 1, 101, 50
Do300i3 = 1, 101, 50
ii = ii + 1
x(ii, 1) = float(i1 - 51) / 100.
x(ii, 2) = float(i2 - 51) / 100.
X(II,3) = FLOAT(I3-51)/100.

300 CONTINUE
GOTO701

- 401 CONTINUE
II=0
DO400I1=1,101,50
DO400I2=1,101,50
DO400I3=1,101,50
DO400I4=1,101,50
II=II+1
X(II,1) = FLOAT(I1-51)/100.
X(II,2) = FLOAT(I2-51)/100.
X(II,3) = FLOAT(I3-51)/100.
X(II,4) = FLOAT(I4-51)/100.

400 CONTINUE
GOTO701

- 501 CONTINUE
II=0
DO500I1=1,101,50
DO500I2=1,101,50
DO500I3=1,101,50
DO500I4=1,101,50
DO500I5=1,101,50
II=II+1
X(II,1) = FLOAT(I1-51)/100.
X(II,2) = FLOAT(I2-51)/100.
X(II,3) = FLOAT(I3-51)/100.
X(II,4) = FLOAT(I4-51)/100.
X(II,5) = FLOAT(I5-51)/100.

500 CONTINUE
GOTO701

- C 601 CONTINUE
II=0
DO600I1=1,101,50
DO600I2=1,101,50
DO600I3=1,101,50
DO600I4=1,101,50
DO600I5=1,101,50
DO600I6=1,101,50
II=II+1
X(II,1) = FLOAT(I1-51)/100.
X(II,2) = FLOAT(I2-51)/100.
X(II,3) = FLOAT(I3-51)/100.
X(II,4) = FLOAT(I4-51)/100.
X(II,5) = FLOAT(I5-51)/100.
X(II,6) = FLOAT(I6-51)/100.

600 CONTINUE
GOTO701

- C 701 CONTINUE
ENTER REST OF POINTS IN THE STAR FORMATION

- C
DO702I=1,N
II=II+1

- C
DO703J=1,N
703 X(II,J)=0.
X(II,I)=1.

702 CONTINUE
DO704I=1,N
II=II+1
DO705 J=1,N
705 X(II,J)=0.
X(II,I)=-1.
704 CONTINUE

ENTER TERMS TO CALCULATE COEFFICIENT ASSOCIATED WITH THE TERM
X(I)**3*X(J)

NMI=N-I
IDO=N-I
J=1
JJ=2
803 CONTINUE
DO804 I=1,IDO
II=II+1
X(II,J)=1.
X(II,JJ)=.5
II=II+1
X(II,J)=.5
JJ=JJ+1
804 CONTINUE
IDO=IDO-I
J=J+1
JJ=J+1
IF (J.LE.NMI) GOTO803

IF WE GOT HERE WE HAVE DEVELOPED THE MINIMUM POINT DESIGN
WRITE(6,*) 'WE HAVE GENERATED ',II,' POINTS IN THE MIN PT DESIGN'
WRITE(7,*) 'WE HAVE GENERATED ',II,' POINTS IN THE MIN PT DESIGN'
WRITE(6,*) 'DESIGN POINTS WRITTEN TO FILE design4.res'

DEVELOP DESIGN POINTS TO AUGMENT THE MINIMUM POINT DESIGN
READ IN THE NUMBER OF RANDOM DESIGN POINTS TO BE DEVELOPED
WRITE(6,*)'ENTER THE NUMBER OF RANDOM GENERATED DESIGN PTS',
X' DESIRED=M'
READ(5,*) M
WRITE(6,*)'NUMBER OF RANDOM DESIGN POINTS M=',M
WRITE(7,*)'NUMBER OF RANDOM DESIGN POINTS M=',M
WRITE(6,*) 'IFLAG IS ANY POSITIVE INTEGER USED TO START RANDOM',
X' PROCESS'
WRITE(6,*)'ENTER IFLAG'
READ(5,*) IFLAG
WRITE(6,*)'IFLAG=',IFLAG
WRITE(7,*)'IFLAG=',IFLAG
DO850 I=1,M
II=II+1
DO851 J=1,N
IFLAG=IFLAG+1
XDUM=RAND(IFLAG)
X(II,J)=2.*XDUM-1.
851 CONTINUE
850 CONTINUE

IF WE GOT HERE WE HAVE FINISHED GENERATING THE RANDOM DESIGN PTS
WRITE(6,*)'RANDOM DESIGN POINTS WRITTEN TO FILE design4.res'

PRINT OUT THE MINIMUM POINT MATRIX IN LOCAL COORDINATES
WRITE(7,*)' DESIGN MATRIX IN LOCAL COORDINATES'
ITOTAL=II
DO700I=1,ITOTAL
WRITE(7,1)I,(X(I,J),J=1,N)
700 CONTINUE
SEE IF WE ARE TO GENERATE DESIGNS IN GLOBAL COORDINATES

WRITE(6,*)' ITEST=1 IF DESIGN POINTS ARE TO BE IN GLOBAL',X' COORDINATES'
WRITE(6,*)' OTHERWISE, ITEST=0'
WRITE(6,*)' ENTER ITEST'
READ(5,*)ITEST
IF(ITEST.NE.1)GOTO860
IF WE GOT HERE WE ARE TO GENERATE DESIGNS IN GLOBAL COORDINATES

WRITE(6,*)' ENTER LOWER AND UPPER RANGE ON EACH DESIGN VARIABLE'
WRITE(6,*)' i.e. ENTER XBB(I) TO XBE(I)'
DO861I=1,N
READ (5, *) XBB(I), XBE(I)
WRITE(6,*)' I,XBB(I),XBE(I)=',I,XBB(I),xBE(I)
WRITE(7,*)' I,XBB(I),XBE(I)=',I,XBB(I),xBE(I)
861 CONTINUE
GOTO862

IF WE GOT HERE LOWER BOUND VARIABLE IN GLOBAL COORDINATES IS -1
IF WE GOT HERE UPPER BOUND VARIABLE IN GLOBAL COORDINATES IS 1
DO863I=1,N
XBB(I)=-1.
XBE(I)=1.
863 CONTINUE

WRITE(7,*)' I,XBB(I),XBE(I),A(I),B(I)
DO1301I=1,N
A(I)=(XBE(I)-XBB(I))/2.
B(I)=(XBE(I)+XBB(I))/2.
WRITE(7,*)I,XBB(I),XBE(I),A(I),B(I)
1301 CONTINUE

DO1202I=1,ITOTAL
DO1202J=1,N
1202 X(I,J)=A(J)*X(I,J)+B(J)
WRITE(6,*)' DESIGN IN GLOBAL COORDINATES WRITEN TO design4.res'
WRITE(6,*)' DESIGN IN GLOBAL COORDINATES WRITEN TO design4.run'
WRITE(7,*)' DESIGN IN GLOBAL COORDINATES'
DO970I=1,ITOTAL
WRITE(7,1)I,(X(I,J),J=1,N)
WRITE(8,11)(X(I,J),J=1,N)
970 CONTINUE
STOP
END
Appendix 3

Program NEWPSI

PROGRAM newpsi

+---------------------------------------------------------------------+
| the program develops a polynomial approximation which                |
| may be either under, exactly, or over determined                     |
| it can handle up to 15 design variables as programmed.              |
| The order of polynomial it can handle is as follows:               |
| 1. one one design variable, up to a 20th order polynomial          |
| 2. two design variables, up to 5th order polynomial                |
| 3. for 2-15 design variables, a 2nd order polynomial               |
| One can use up to 250 designs to train the approximation.          |
| It can handle up to 2000 grid points                               |
+---------------------------------------------------------------------+

IMPLICIT REAL*8 (A-H,O-Z)
dimension x(250,15),y(250),a(250,136)
dimension yhat(250)
dimension b(136)
dimension yyhat(2000)

1 FORMAT (9F8.4)
2 FORMAT (3F12.6)
3 FORMAT (F10.6,1H,,F10.6,1H,,F10.6,1H,,F10.6,1H,,F10.6,1H,,F10.6,
  X1H,,F10.6)
OPEN (UNIT=5,FILE='newpsi.dat',)
OPEN (UNIT=7,FILE='newpsi.res',)
OPEN (UNIT=8,FILE='newpsi.plot')

+---------------------------------------------------------------------+
| read in data                                                        |
+---------------------------------------------------------------------+
read in the print code
read(5,*)ip

enter number of design variables, nd
read(5,*)nd

enter THE DEGREE OF POLYNOMIAL TO BE CONSIDERED, np
READ(5,*)np

ENTER NUMBER OF DESIGNS FOR PROBLEM,M
READ(5,*)M

write(6,*)' print code ip=',ip
write(6,*)' number of design variables, nd=',nd
write(6,*)' degree of polynomial being considered=np=',np
write(6,*)' number of designs m=',m
write(7,*)' print code ip=',ip
write(7,*)' number of design variables, nd=',nd
write(7,*)' degree of polynomial being considered=np=',np
write(7,*)' number of designs m=',m

read in designs and set up matrix a

write(7,*)' x(i,j), y(i)'
DO101I=1,M
read(5,*) (x(i,j), j=1,nd), y(i)
write(7,*) (x(i,j), j=1,nd), y(i)
101 continue

set up the coefficient matrix, a, in the matrix equation
y=a x
call geta(ip,m,nd,np,n,x,a)

SEE WHETHER SYSTEM IS UNDER, EXACTLY, OR OVER DETERMINED

IF(M.GE.N)GOTO400
IF WE GOT HERE WE ARE UNDER-DETERMINED
WRITE(6,*)' SYSTEM IS UNDER-DETERMINED'
WRITE(7,*)' SYSTEM IS UNDER-DETERMINED'
CALL PSI(ip,M,N,A,Y,B)
GOTO402
400 CONTINUE

IF(M.GT.N)GOTO401
IF WE GOT HERE WE ARE EXACTLY DETERMINED
WRITE(6,*)' SYSTEM IS EXACTLY DETERMINED'
WRITE(7,*)' SYSTEM IS EXACTLY DETERMINED'
CALL EXACT(ip,M,A,Y,B)
GOTO402
401 CONTINUE

IF WE GOT HERE WE ARE OVER-DETERMINED
WRITE(6,*)' SYSTEM IS OVER-DETERMINED'
WRITE(7,*)' SYSTEM IS OVER-DETERMINED'
CALL OVER(ip,M,N,A,Y,B)
402 CONTINUE

EVALUATE APPROXIMATION AT DESIGNS

WRITE(6,*)' MATRIX OF COEFFICIENTS, B(I)'
WRITE(7,*)' MATRIX OF COEFFICIENTS, B(I)'
WRITE(6,*) (B(I), I=1,N)
WRITE(7,*) (B(I), I=1,N)
WRITE(7,*)' MATRICES Y(I) AND YHAT(I)'

recalculate matrix a
call geta(ip,m,nd,np,n,x,a)
calculate approximation at designs and print results

write(7,*)' y(i), yhat(i)'
DO102I=1,M
YHAT(I)=0.
DO103J=1,N
yhat(i) = yhat(i) + a(i,j) * b(j)

CONTINUE
WRITE(7,*) Y(I), YHAT(I)

CONTINUE

evaluate function at grid
read(5,*) ng
write(6,*)' number of designs on grid = ng', ng
write(7,*)' number of designs on grid = ng', ng
write(7,*)' xx(i,j), yy(i)'
DO601I=1,ng
read(5,*) (xx(i,j), j=1, nd), yy(i)
write(7,*) (xx(i,j), j=1, nd), yy(i)

CONTINUE

DO601 continue
call getabg(ip, ng, nd, np, n, xx, abig)
write(7,*)' yy(i), yyhat(i) at grid'
DO602I=1, ng
YYHAT(I)=0.
DO603J=1, N
yyhat(i) = yyhat(i) + abig(i,j) * b(j)

CONTINUE

WRITE(7,*) YY(I), YYHAT(I)

write the plot file
write(8,*) (xx(i,j), j=1, nd), yyhat(i)

CONTINUE

calculate statistical terms

call statit(m, y, yhat, ng, yy, yyhat)

STOP
END

subroutine geta(ip, m, nd, np, n, x, a)

This subroutine generates the matrix a where the matrix equation is y = a b. Here y are the training functions, b are undetermined coefficients. The algorithm is programmed to handle
1. any level of approximation for one design variable
2. up to 5th order polynomial in two design variables
3. quadratic approximation in more than two design variables

IMPLICIT REAL*8 (A-H,O-Z)
dimension x(250,15), a(250,136)

do for each design

do300i=1, m

if nd is not equal to 1 go to 400
if (nd.ne.1) goto 400

here we have nd=1, i.e. one design variable
we will develop a's for all np's

    a(i,1)=1.
    j=1
    do201 k=1,np
          j=j+1
          a(i,j)=x(i,1)**k
    201 continue
    n=np+1
    goto 301

400 continue

if nd is not equal to 2 go to 500
if (nd.ne.2) goto 500

if we got here we have 2 design variables

    x1=x(i,1)
    x2=x(i,2)

add the constant and linear terms

    a(i,1)=1.
    a(i,2)=x1
    a(i,3)=x2
    n=3
    if (np.lt.2) goto 301

add the 2nd order terms

    a(i,4)=x1**2
    a(i,5)=x1*x2
    a(i,6)=x2**2
    n=6
    if (np.lt.3) goto 301

add the cubic terms

    a(i,7)=x1**3
    a(i,8)=x1**2*x2
    a(i,9)=x1*x2**2
    a(i,10)=x2**3
n=10
if(np.lt.4)goto301

**************************
add the 4th order terms

   a(i,11)=x1**4
   a(i,12)=x1**3*x2
   a(i,13)=x1**2*x2**2
   a(i,14)=x1*x2**3
   a(i,15)=x2**4
n=15
if(np.lt.5)goto301

**************************
add the 5th order terms

   a(i,16)=x1**5
   a(i,17)=x1**4*x2
   a(i,18)=x1**3*x2**2
   a(i,19)=x1**2*x2**3
   a(i,20)=x1*x2**4
   a(i,21)=x2**5
n=21
if(np.lt.6)goto301

**************************
algorithm not programed for polynomials of order larger than 5

write(6,*)' for two design variables, algorithm not programed for'
write(6,*)' polynomials of order larger than 5'
write(7,*)' for two design variables, algorithm not programed for'
write(7,*)' polynomials of order larger than 5'
stop

********************

500 continue

if we got here number of design variables >2

**************************

enter constant and linear terms

   a(i,1)=1.
   j=1
d0501k=1,nd
   j=j+1
   a(i,j)=x(i,k)
501 continue
   n=j
if(np.lt.2)goto301

**************************
enter the quadratic terms

do502k=1, nd
do502L=k, nd
j=j+1
a(i, j) = x(i,k) * x(i, L)

502 continue
n=j
if(np. lt. 3) goto301

algorithm not programmed for more than quadratic approximation when number of design variables >2

write(6,*), ' algorithm not programmed for more than quadratic'
write(6,*), ' approximation when number of design variables >2'
write(7,*), ' algorithm not programmed for more than quadratic'
write(7,*), ' approximation when number of design variables >2'
stop

print out some results

301 continue
if(ip. lt. 4) goto302
write(6,*), ' a(i,j)', (a(i,j), j=1,n)
write(6,*), ' a(i,j)', (a(i,j), j=1,n)
write(7,*), ' a(i,j)', (a(i,j), j=1,n)
write(7,*), ' a(i,j)', (a(i,j), j=1,n)

302 continue

print out some results

300 continue
write(6,*), ' number of undetermined coef=n=', n
write(7,*), ' number of undetermined coef=n=', n
return
end subroutine getabg(ip, m, nd, np, n, x, a)

This subroutine generates the matrix a where the matrix equation is \( y = a \cdot b \). Here \( y \) are the training functions, \( b \) are undetermined coefficients. The algorithm is programmed to handle
1. any level of approximation for one design variable
2. up to 5th order polynomial in two design variables
3. quadratic approximation in more than two design variables

IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION A(2000,136)
DIMENSION X(2000,15)
do for each design

   do300i=1,m

   **********************************************

   if nd is not equal to 1 go to 400
   if(nd.ne.1)goto400

   **********************************************
   **********************************************

   here we have nd=1, i.e. one design variable
   we will develop a's for all np's

   a(i,1)=1.
   j=1
   do201k=1,np
   j=j+1
   a(i,j)=x(i,1)**k
   continue
   n=np+1
   goto301

   **********************************************

   400 continue

   if nd is not equal to 2 go to 500
   if(nd.ne.2)goto500

   **********************************************
   **********************************************

   if we got here we have 2 design variables

   xl=x(i,1)
x2=x(i,2)

   ***********************

   add the constant and linear terms

   a(i,1)=1.
a(i,2)=xl
   a(i,3)=x2
   n=3
   if(np.lt.2)goto301

   *************************

   add the 2nd order terms

   a(i,4)=xl**2
   a(i,5)=xl*x2
   a(i,6)=x2**2
   n=6
   if(np.lt.3)goto301
add the cubic terms

\[
a(i,7) = x_1^3
\]
\[
a(i,8) = x_1^2 x_2
\]
\[
a(i,9) = x_1 x_2^2
\]
\[
a(i,10) = x_2^3
\]
n=10
if(np.lt.4)goto301

add the 4th order terms

\[
a(i,11) = x_1^4
\]
\[
a(i,12) = x_1^3 x_2
\]
\[
a(i,13) = x_1^2 x_2^2
\]
\[
a(i,14) = x_1 x_2^3
\]
\[
a(i,15) = x_2^4
\]
n=15
if(np.lt.5)goto301

add the 5th order terms

\[
a(i,16) = x_1^5
\]
\[
a(i,17) = x_1^4 x_2
\]
\[
a(i,18) = x_1^3 x_2^2
\]
\[
a(i,19) = x_1^2 x_2^3
\]
\[
a(i,20) = x_1 x_2^4
\]
\[
a(i,21) = x_2^5
\]
n=21
if(np.lt.6)goto301

algorithm not programed for polynomials of order larger than 5

write(6,*)' for two design variables, algorithm not programed for'
write(6,*)' polynomials of order larger than 5'
write(7,*)' for two design variables, algorithm not programed for'
write(7,*)' polynomials of order larger than 5'
stop

continue

if we got here number of design variables >2

enter constant and linear terms

\[
a(i,1) = 1.
\]
j=1
do501k=1,nd
  j=j+1
  a(i,j)=x(i,k)
  continue
n=j
  if(np.lt.2)goto301

***************************
enter the quadratic terms
do502k=1,nd
do502L=k,nd
  j=j+1
  a(i,j)=x(i,k)*x(i,L)
  continue
n=j
  if(np.lt.3)goto301

***************************
algorithm not programmed for more than quadratic approximation
when number of design variables >2
write(6,*)' algorithm not programmed for more than quadratic'
write(6,*)' approximation when number of design variables >2'
write(7,*)' algorithm not programmed for more than quadratic'
write(7,*)' approximation when number of design variables >2'
stop

***************************************************************
print out some results
301 continue
  if(ip.lt.4)goto302
  write(6,*)' a(i,j)',(a(i,j),j=1,n)
  write(6,*)'
  write(7,*)' a(i,j)',(a(i,j),j=1,n)
  write(7,*)'
302 continue

***************************************************************
300 continue
  write(6,*)' number of undetermined coef=n=',n
  write(7,*)' number of undetermined coef=n=',n
return
end
SUBROUTINE PSI(IP,M,N,DUMA,Y,XX)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION DUMA(250,136)
DIMENSION A(21,21),B(21,21),D(21,21),DI(21,21),BPI(21,21)
DIMENSION C(21,21),FI(21,21),CPI(21,21),H(21,21),HI(21,21)
DIMENSION API(21,21)
DIMENSION F(21,21)
DIMENSION IPIVOT(21),IWK(21,2)
DIMENSION y(250)
DIMENSION XX(21)

THIS SUBROUTINE CALCULATES PSEUDO INVERSE OF MATRIX A
M = ROW DIMENSION OF A LESS THAN N
N = COLUMN DIMENSION OF A

COPY DUMA TO A

DO90I=1,M
DO90J=1,N
90 A(I,J)=DUMA(I,J)

PRINT MATRIX A
if(ip.lt.4)goto50
WRITE(6,*)' MATRIX A'
WRITE(7,*)' MATRIX A'
call writit(m,n,a)
50 continue

SET UP MATRIX B

DO100I=1,M
DO100J=1,M
100 B(I,J)=A(I,J)
if(ip.lt.4)goto51
WRITE(6,*)' MATRIX B'
WRITE(7,*)' MATRIX B'
call writit(m,m,b)
51 continue

GET D = B TRAN * B

DO101I=1,M
DO101J=1,M
D(I,J)=0.
DO101K=1,M
101 D(I,J)=D(I,J)+B(K,I)*B(K,J)
if(ip.lt.4)goto52
WRITE(6,*)' MATRIX D'
WRITE(7,*)' MATRIX D'
call writit(m,m,d)
52 continue

GET INVERSE OF D=DI
MAX=21
MDUM=0
IOP=0
call matinv(max,m,d,mdum,di,iop, determ,iscale,ipivot,iwk)
WRITE(6,*)' DETERM=',determ,' ISCALE=',iscale
WRITE(7,*)' DETERM=',determ,' ISCALE=',iscale
do300i=1,m
do300j=1,m
300 DI(I,J)=D(I,J)
if(ip.lt.4)goto53
WRITE(6,*)' MATRIX DI'
WRITE(7,*)' MATRIX DI'
call writit(m,m,di)
53 continue
GET PSEUDO INVERSE OF B = BPI = DI * B TRANS

DO102I=1,M
DO102 JQ=1,M
BPI(I,JQ)=0.
DO102J=1,M
102 BPI(I,JQ)=BPI(I,JQ)+DI(I,J)*B(JQ,J)
if(ip.lt.4)goto54
WRITE(6,*)' MATRIX BPI'
WRITE(7,*)' MATRIX BPI'
CALL WRITIT(M,M,BPI)
continue

SET UP MATRIX C = A

DO103I=1,M
DO103J=1,N
103 C(I,J)=A(I,J)
if(ip.lt.4)goto55
WRITE(6,*)' MATRIX C'
WRITE(7,*)' MATRIX C'
CALL WRITIT(M,N,C)
continue

SET UP MATRIX F = C * C TRANS

DO104I=1,M
DO104J=1,M
F(I,J)=0.
DO104K=1,N
104 F(I,J)=F(I,J)+C(I,K)*C(J,K)
if(ip.lt.4)goto56
WRITE(6,*)' MATRIX F'
WRITE(7,*)' MATRIX F'
CALL WRITIT(M,M,F)
continue

GET THE INVERSE OF F = FI

CALL MATINV(MAX,M,F,MDUM,FI,IOP,DETERM,ISCALE,IPIVOT,IWK)
WRITE(6,*)' DETERM=',DETERM
WRITE(7,*)' DETERM=',DETERM
WRITE(6,*)' ISCALE=',ISCALE
WRITE(7,*)' ISCALE=',ISCALE
DO301I=1,M
DO301J=1,M
301 FI(I,J)=F(I,J)
if(ip.lt.4)goto57
WRITE(6,*)' MATRIX FI'
WRITE(7,*)' MATRIX FI'
CALL WRITIT(M,M,FI)
continue

GET THE PSEUDO INVERSE OF C = CPI = C TRANS * FI

DO105IQ=1,N
DO105J=1,M
CPI(IQ,J)=0.
DO105I=1,M
105 CPI(IQ,J)=CPI(IQ,J)+C(I,IQ)*FI(I,J)
if(ip.lt.4)goto58
C

WRITE(6,*)' MATRIX CPI'
WRITE(7,*)' MATRIX CPI'
CALL WRITIT(N,M,CPI)

58 continue

SET UP MATRIX H = PSEUDO INVERSE OF B = BPI

DO106I=1,M
DO106J=1,M
106 H(I,J)=BPI(I,J)
    if(ip.lt.4)goto59
    WRITE(6,*)' MATRIX H'
    WRITE(7,*)' MATRIX H'
    CALL WRITIT(M,M,H)

59 continue

GET INVERSE OF H = HI
    CALL MATINV(MAX,M,H,MDUM,H,IOP,DETERM,ISCALE,IPIVOT,IWK)
    WRITE(6,*)' DETERM=',DETERM,' ISCALE=',ISCALE
    WRITE(7,*)' DETERM=',DETERM,' ISCALE=',ISCALE
    DO302I=1,M
    DO302J=1,M
302 HI(I,J)=H(I,J)
    if(ip.lt.4)goto60
    WRITE(6,*)' MATRIX HI'
    WRITE(7,*)' MATRIX HI'
    CALL WRITIT(M,M,HI)

60 continue

GET PSEUDO INVERSE OF A = API = CPI * HI * BPI

DO107IQ=1,N
DO107J=1,M
API(IQ,J)=0.
    DO107I=1,M
    DO107K=1,M
107 API(IQ,J)=API(IQ,J)+CPI(IQ,I)*HI(I,K)*BPI(K,J)
    if(ip.lt.4)goto61
    WRITE(6,*)' MATRIX API'
    WRITE(7,*)' MATRIX API'
    CALL WRITIT(N,M,API)

61 continue

GET XX = API * Y

DO108IQ=1,N
XX(IQ)=0.
    DO108J=1,M
108 XX(IQ)=XX(IQ)+API(IQ,J)*Y(J)
    JDUM=1
    if(ip.lt.4)goto62
    WRITE(6,*)' MATRIX XX'
    WRITE(7,*)' MATRIX XX'
    CALL WRITIT(N,JDUM,XX)

62 continue

RETURN

END

SUBROUTINE WRITIT(MM,NN,XX)
IMPLICIT REAL*8 (A-H,O-Z)
DIMENSION XX(21,1)
1 FORMAT (1X)
2 FORMAT (10F7.2)
WRITE (6, 1)
DO100 I = 1, MM
WRITE (6, 2) (XX(I,J), J = 1, NN)
WRITE (7, 2) (XX(I,J), J = 1, NN)
100 CONTINUE
RETURN
END

SUBROUTINE EXACT(IP, M, A, Y, B)
IMPLICIT REAL*8 (A-H, O-Z)

DIMENSION a(250, 136), b(136), y(250)
DIMENSION IPIVOT(250), IWK(250, 2)
DIMENSION C(136, 1)
DO100 I = 1, M
100 C(I, 1) = Y(I)
MAX = 250
MDUM = 1
IOP = 0
CALL MATINV(MAX, M, A, MDUM, C, IOP, DETERM, ISCALE, IPIVOT, IWK)
WRITE (6, *) ' DETERM = ' , DETERM, ' ISCALE = ' , ISCALE
WRITE (7, *) ' DETERM = ' , DETERM, ' ISCALE = ' , ISCALE
DO101 I = 1, M
B(I) = C(I, 1)
101 CONTINUE
if (IP .LT. 4) goto 50
WRITE (6, *) ' MATRIX B', (B(I), I = 1, M)
WRITE (7, *) ' MATRIX B', (B(I), I = 1, M)
continue
RETURN
END

SUBROUTINE OVER(IP, M, N, A, Y, B)
IMPLICIT REAL*8 (A-H, O-Z)

DIMENSION a(250, 136), b(136), y(250)
DIMENSION IPIVOT(136), IWK(136, 2)
DIMENSION ATA(136, 136), ATY(136, 1)
DO200 I = 1, N
DO200 J = 1, N
ATA(I, J) = 0.
DO200 K = 1, M
ATA(I, J) = ATA(I, J) + A(K, I) * A(K, J)
DO201 I = 1, N
ATY(I, 1) = 0.
DO201 K = 1, M
200 ATY(I, J) = ATY(I, J) + A(K, I) * Y(K)
MAX = 136
MDUM = 1
IOP = 0
CALL MATINV(MAX, N, ATA, MDUM, ATY, IOP, DETERM, ISCALE, IPIVOT, IWK)
WRITE (6, *) ' DETERM = ' , DETERM, ' ISCALE = ' , ISCALE
WRITE (7, *) ' DETERM = ' , DETERM, ' ISCALE = ' , ISCALE
DO101 I = 1, N
B(I) = ATY(I, 1)
101 CONTINUE
if (IP .LT. 4) goto 50
WRITE (6, *) ' MATRIX B', (B(I), I = 1, N)
WRITE (7, *) ' MATRIX B', (B(I), I = 1, N)
continue
This subroutine calculates quality of approximation measures.

** dimension y(250), yhat(250) 

yb=0.

do100 id=1,m 
  yb=yb+y(id)
100 continue 
  yb=yb/float(m) 
error=0.

do101 id=1,m 
  error=error+(y(id)-yhat(id))**2
101 continue 
  v=sqrt(error/float(m))/yb*(100.)
write(7,*)' error over designs=error = ',error 
write(7,*)' average y over design = yb =',yb 
write(6,*)' coefficient v (as %)= ',v 
write(7,*)' coefficient v (as %)= ',v 

dn=0.
dd=0.

do7769 id=1,m 
  dn=dn+(yhat(id)-yb)**2
7769 continue 
  dd=dd+(y(id)-yb)**2 
  r2=dn/dd*(100.) 
write(6,*)' coefficient r2 (as%) = ',r2 
write(7,*)' coefficient r2 (as%) = ',r2 

give vg 

perror=0.

gyg=0.

do155 id=1,ng 
  yg=yg+yy(id) 
  perror=perror+(yy(id)-yyhat(id))**2 
155 continue 
  yg=yg/float(ng) 
vg=sqrt(perror/float(ng))/yg*(100.) 
write(7,*)' sum of residuals squared=perror=',perror 
write(7,*)' average y over grid = yg =',yg 
write(6,*)' coefficient vg = ',vg 
write(7,*)' coefficient vg = ',vg 
return 
end 

SUBROUTINE MATINV(MAX,N,A,M,B,IOP,DETERM,ISCALE,IPIVOT,IWK) 

implicit real*8 (a-h,o-z) 

C FI.3 MATINV 3 
C PURPOSE - MATINV INVERTS A REAL SQUARE MATRIX A. 
C IN ADDITION THE ROUTINE SOLVES THE MATRIX 

MATINV 2
EQUATION AX=B, WHERE B IS A MATRIX OF CONSTANT VECTORS. THERE IS ALSO AN OPTION TO HAVE THE DETERMINANT EVALUATED. IF THE INVERSE IS NOT NEEDED, USE GELIM TO SOLVE A SYSTEM OF SIMULTANEOUS EQUATIONS AND DETFAC TO EVALUATE A DETERMINANT FOR SAVING TIME AND STORAGE.

USE - CALL MATINV(MAX,N,A,M,B,IOP,DETERM,ISCALE,IPIVOT,IWK)

MAX - THE MAXIMUM ORDER OF A AS STATED IN THE DIMENSION STATEMENT OF THE CALLING PROGRAM.

N - THE ORDER OF A, 1.LE.N.LE.MAX.

A - A TWO-DIMENSIONAL ARRAY OF THE COEFFICIENTS. ON RETURN TO THE CALLING PROGRAM, A INVERSE IS STORED IN A. A MUST BE DIMENSIONED IN THE CALLING PROGRAM WITH FIRST DIMENSION MAX AND SECOND DIMENSION AT LEAST N.

M - THE NUMBER OF COLUMN VECTORS IN B. M=0 SIGNALS THAT THE SUBROUTINE IS USED SOLELY FOR INVERSION, HOWEVER, IN THE CALL STATEMENT AN ENTRY CORRESPONDING TO B MUST BE PRESENT.

B - A TWO-DIMENSIONAL ARRAY OF THE CONSTANT VECTOR B. ON RETURN TO CALLING PROGRAM, X IS STORED IN B. B SHOULD HAVE ITS FIRST DIMENSION MAX AND ITS SECOND AT LEAST M.

IOP - COMPUTE DETERMINANT OPTION.
   IOP=0 COMPUTES THE MATRIX INVERSE AND DETERMINANT.
   IOP=1 COMPUTES THE MATRIX INVERSE ONLY.

DETERM - FOR IOP=0- IN CONJUNCTION WITH ISCALE REPRESENTS THE VALUE OF THE DETERMINANT OF A, DET(A), AS FOLLOWS.
   DET(A) = (DETERM) (10**100(ISCALE))
   THE COMPUTATION DET(A) SHOULD NOT BE ATTEMPTED IN THE USER PROGRAM SINCE IF THE ORDER OF A IS LARGER AND/OR THE MAGNITUDE OF ITS ELEMENTS ARE LARGE(SMALL), THE DET(A) CALCULATION MAY CAUSE OVERFLOW (UNDERFLOW). DETERM SET TO ZERO FOR SINGULAR MATRIX CONDITION, FOR EITHER IOP=1, OR 0. SHOULD BE CHECKED BY PROGRAMER ON RETURN TO MAIN PROGRAM.

ISCALE - A SCALE FACTOR COMPUTED BY THE SUBROUTINE TO AVOID OVERFLOW OR UNDERFLOW IN THE COMPUTATION OF THE QUANTITY, DETERM.

IPIVOT - A ONE DIMENSIONAL INTEGER ARRAY USED BY THE SUBPROGRAM TO STORE PIVOTOL INFORMATION. IT SHOULD BE DIMENSIONED AT LEAST N. IN GENERAL
THE USER DOES NOT NEED TO MAKE USE
OF THIS ARRAY.

IWK - A TWO-DIMENSIONAL INTEGER ARRAY OF
TEMPORARY STORAGE USED BY THE ROUTINE.
IWK SHOULD HAVE ITS FIRST DIMENSION
MAX, AND ITS SECOND 2.

REQUIRED ROUTINES-

REFERENCE - FOX, L, AN INTRODUCTION TO NUMERICAL
LINEAR ALGEBRA

STORAGE - 542 OCTAL LOCATIONS

LANGUAGE - FORTRAN

LIBRARY FUNCTIONS - ABS

RELEASED - JULY 1973

LATEST REVISION - JULY 29, 1981

COMPUTER SCIENCES CORPORATION
HAMPTON, VA

**Initialization**

ISCALE = 0
R1 = (10.0d+00)**32
R2 = 1.0d+00/R1
DETERM = 1.0d+00
DO 20 J = 1, N
   IPIVOT (J) = 0
   CONTINUE
DO 550 I = 1, N

**Search for Pivot Element**

AMAX = 0.0d+00
DO 105 J = 1, N
   IF (IPIVOT (J) - 1) 60, 105, 60
   DO 100 K = 1, N
      IF (IPIVOT (K) - 1) 80, 100, 740
      TMAX = ABS (A (J,K))
      IF (AMAX - TMAX) 85, 100, 100
      IROW = J
      ICOLUMN = K
      AMAX = TMAX
   CONTINUE
100 CONTINUE
105 CONTINUE
IF (AMAX) 740, 106, 110
DETERM = 0.0d+00
ISCALE = 0
GO TO 740
110 IPIVOT (ICOLUMN) = 1

**Interchange Rows to Put Pivot Element on Diagonal**
IF (IROW-ICOLUM) > 140, 260, 140
DETERM=-DETERM
DO 200 L=1,N
    SWAP=A(IROW,L)
    A(IROW,L)=A(ICOLUM,L)
    A(ICOLUM,L)=SWAP
200 CONTINUE
IF(M) 260, 260, 210
DO 250 L=1,M
    SWAP=B(IROW,L)
    B(IROW,L)=B(ICOLUM,L)
    B(ICOLUM,L)=SWAP
250 CONTINUE
IWK(I,1)=IROW
IWK(I,2)=ICOLUM
PIVOT=A(ICOLUM,ICOLUM)
IF(IOP) 740,1000,321
C C
SCALE THE DETERMINANT
C
1000 PIVOTI=PIVOT
   IF(ABS(DETERM)-R1) > 1030,1010,1010
   1010 DETERM=DETERM/R1
   ISCALE=ISCALE+1
   IF(ABS(DETERM)-R1) > 1060,1020,1020
   1020 DETERM=DETERM/R1
   ISCALE=ISCALE+1
   GO TO 1060
   1030 IF(ABS(DETERM)-R2) > 1040,1040,1060
   1040 DETERM=DETERM*R1
   ISCALE=ISCALE-1
   IF(ABS(DETERM)-R2) > 1050,1050,1060
   1050 DETERM=DETERM*R1
   ISCALE=ISCALE-1
   1060 IF(ABS(PIVOTI)-R1) > 1090,1070,1070
   1070 PIVOTI=PIVOTI/R1
   ISCALE=ISCALE+1
   IF(ABS(PIVOTI)-R1) > 320,1080,1080
   1080 PIVOTI=PIVOTI/R1
   ISCALE=ISCALE+1
   GO TO 320
   1090 IF(ABS(PIVOTI)-R2) > 2000,2000,320
   2000 PIVOTI=PIVOTI*R1
   ISCALE=ISCALE-1
   IF(ABS(PIVOTI)-R2) > 2010,2010,320
   2010 PIVOTI=PIVOTI*R1
   ISCALE=ISCALE-1
   320 DETERM=DETERM*PIVOTI
C C
DIVIDE PIVOT ROW BY PIVOT ELEMENT
C
321 A(ICOLUM,ICOLUM)=1.0d+00
   DO 350 L=1,N
       A(ICOLUM,L)=A(ICOLUM,L)/PIVOT
   350 IF(M) 380, 380, 360
   360 DO 370 L=1,M
       B(ICOLUM,L)=B(ICOLUM,L)/PIVOT
   370 C C
REDUCE NON-PIVOT ROWS
C
C
C
C
C

380 DO 550 L1=1,N
  IF(L1-ICOLUM) 400, 550, 400
   T=A(L1,ICOLUM)
   A(L1,ICOLUM)=0.0d+00
   DO 450 L=1,N
   A(L1,L)=A(L1,L)-A(ICOLUM,L)*T
   IF(M) 550, 550, 460
   DO 450 L=1,M
   B(L1,L)=B(L1,L)-B(ICOLUM,L)*T
550 CONTINUE
INTERCHANGE COLUMNS
DO 710 I=1,N
  L=N+I-I
  IF (IWK(L,1)-IWK(L,2)) 630, 710, 630
  JROW=IWK(L,1)
  JCOLUMN=IWK(L,2)
  DO 705 K=1,N
  SWAP=A(K,JROW)
  A(K,JROW)=A(K,JCOLUMN)
  A(K,JCOLUMN)=SWAP
705 CONTINUE
710 CONTINUE
740 RETURN
END

ROUTINE NAME  - HC318=EPSLON
FROM EISPACK

LATEST REVISION  - AUGUST 1,1984
       COMPUTER SCIENCES CORP., HAMPTON, VA.
PURPOSE  - THE FORTRAN FUNCTION EPSLON ESTIMATES UNIT
ROUNDROFF IN QUANTITIES OF SIZE X.
USAGE  - VARIABLE = EPSLON(X)
ARGUMENTS   X  - IS A REAL INPUT VARIABLE WHICH REPRESENTS THE
QUANTITIES OF SIZE IN WHICH UNIT ROUNDROFF
WILL BE ESTIMATED.
REQUIRED ROUTINES  - NONE
REMARKS  1.  IT SHOULD BE NOTED THAT EPSLON IS A FUNCTION
DESIGNED TO BE CALLED BY ROUTINES IN THE
EISPACK VERSION 3.

THIS PROGRAM SHOULD FUNCTION PROPERLY ON ALL
SYSTEMS SATISFYING THE FOLLOWING TWO
ASSUMPTIONS,

A.  THE BASE USED IN REPRESENTING FLOATING
POINT NUMBERS IS NOT A POWER OF THREE.

B.  THE QUANTITY A IN STATEMENT 10 IS
REPRESENTED TO THE ACCURACY USED IN FLOATING
POINT VARIABLES THAT ARE STORED IN MEMORY.
THE STATEMENT NUMBER 10 AND THE GO TO 10 ARE
INTENDED TO FORCE OPTIMIZING COMPILERS TO
GENERATE CODE SATISFYING ASSUMPTION 2.

UNDER THESE ASSUMPTIONS, IT SHOULD BE TRUE
THAT,

A IS NOT EXACTLY EQUAL TO FOUR-THIRDS,
B HAS A ZERO FOR ITS LAST BIT OR DIGIT,
C IS NOT EXACTLY EQUAL TO ONE,

EPS MEASURES THE SEPARATION OF 1.0 FROM THE
NEXT LARGER FLOATING POINT NUMBER.

EXAMPLE:

PROGRAM TR(OUTPUT,TAPE6=OUTPUT)
REAL X
X = 4.
A = EPSLON(X)
WRITE(6,100) A
STOP
END

REAL*8 FUNCTION EPSLON (X)
REAL*8 X
REAL*8 A,B,C,EPS
A = 4.0E0/3.0E0
B = A - 1.0E0
C = B + B + B
EPS = ABS(C-1.0E0)
IF (EPS .EQ. 0.0E0) GO TO 10
EPSLON = EPS*ABS(X)
RETURN

THIS PROGRAM VALID ON FTN4 AND FTN5 **

ROUTINE NAME - PF260=QZHES
FROM EISPACK

LATEST REVISION - AUGUST 1,1984
COMPUTER SCIENCES CORP., HAMPTON, VA.

PURPOSE - THIS SUBROUTINE ACCEPTS A PAIR OF REAL
GENERAL MATRICES AND REDUCES ONE OF THEM TO
UPPER HESSENBERG FORM AND THE OTHER TO UPPER
TRIANGULAR FORM USING ORTHOGONAL
TRANSFORMATIONS. IT IS USUALLY FOLLOWED BY
QZIT(PF261), QZVAL(PF262) AND, POSSIBLY,
QZVEC(PF263).
- CALL QZGES(NM,N,A,B,MATZ,Z)

**ARGUMENTS**

**NM**
- On input NM must be set to the row dimension of two-dimensional array parameters as declared in the calling program dimension statement.

**N**
- On input N is the order of the matrices.

**A**
- On input A contains a real general matrix. Must be of dimension NM x N.

- On output A has been reduced to upper Hessenberg form. The elements below the first subdiagonal have been set to zero.

**B**
- On input B contains a real general matrix. Must be of dimension NM x N.

- On output B has been reduced to upper triangular form. The elements below the main diagonal have been set to zero.

**MATZ**
- On input MATZ should be set to .TRUE. if the right hand transformations are to be accumulated for later use in computing eigenvectors, and to .FALSE. otherwise.

**Z**
- On output Z contains the product of the right hand transformations if MATZ has been set to .TRUE. otherwise, Z is not referenced.

Must be of dimension NM x N.

**REQUIRED ROUTINES**
- NONE

**REMARKS**

1. This subroutine is the first step of the QZ algorithm for solving generalized matrix eigenvalue problems, SIAM J. NUMER. ANAL. 10, 241-256(1973) by Moler and Stewart.

**EXAMPLE**

```plaintext
PROGRAM TQZGES(OUTPUT,TAPE6=OUTPUT)
DIMENSION A(5,5),Z(5,5),B(5,5)
LOGICAL MATZ

N = 5
NM = 5
MATZ = .TRUE.

DATA A /10.,2.,3.,2*1.,2.,12.,1.,2.,1.,3.,1.,11.,
* 1.,-1.,1.,2.,1.,9.,3*1.,-1.,1.,15. /

DATA B /12.,1.,-1.,1.,2.*1.,14.,1.,-1.,1.,-1.,1.,
* 16.,-1.,1.,2.,-1.,-1.,1.,-1.,1.,12.,-1.,3*1.,-1.,11. /

CALL QZGES(NM,N,A,B,MATZ,Z)
WRITE(6,100) ((A(I,J),I=1,5),J=1,5), ((B(I,J),I=1,5),J=1,5),
```

QZGES 18
QZGES 19
QZGES 20
QZGES 21
QZGES 22
QZGES 23
QZGES 24
QZGES 25
QZGES 26
QZGES 27
QZGES 28
QZGES 29
QZGES 30
QZGES 31
QZGES 32
QZGES 33
QZGES 34
QZGES 35
QZGES 36
QZGES 37
QZGES 38
QZGES 39
QZGES 40
QZGES 41
QZGES 42
QZGES 43
QZGES 44
QZGES 45
QZGES 46
QZGES 47
QZGES 48
QZGES 49
QZGES 50
QZGES 51
QZGES 52
QZGES 53
QZGES 54
QZGES 55
QZGES 56
QZGES 57
QZGES 58
QZGES 59
QZGES 60
QZGES 61
QZGES 62
QZGES 63
QZGES 64
QZGES 65
QZGES 66
QZGES 67
QZGES 68
QZGES 69
QZGES 70
QZGES 71
QZGES 72
QZGES 73
QZGES 74
QZGES 75
QZGES 76
QZGES 77
SUBROUTINE QZHES(NM,N,A,B,MATZ,Z)

implicit real*8 (a-h,o-z)
INTEGER I,J,K,L,N,LB,LI,NM,NKI,NM2
REAL*8 A(NM,N),B(NM,N),Z(NM,N)
REAL*8 R,S,T,UI,U2,VI,V2,RHO
LOGICAL MATZ
IF (.NOT. MATZ) GO TO 10

DO 3 J = 1, N
  DO 2 I = 1, N
    Z(I,J) = 0.0E0
  CONTINUE

  Z(J,J) = 1.0E0
3 CONTINUE

............ REDUCE B TO Upper Triangular Form ............
10 IF (N .LE. 1) GO TO 170
NM1 = N - 1

DO 100 L = 1, NM1
  L1 = L + 1
  S = 0.0E0

  DO 20 I = L1, N
    S = S + ABS(B(I,L))
  CONTINUE

10 IF (S .EQ. 0.0E0) GO TO 100
  S = S + ABS(B(L,L))

STOP
END

OUTPUT:

A =
-9.9  4.1  0.  0.  0.
-2.4  11. -3.0  0.  0.
  .91  .26 -13.  3.3  0.
-3.8  2.0  1.7 -11.  2.6
  2.7 -1.5  .99  1.4 -11.
B =
  -12.  0.  0.  0.  0.
   2.3  16.  0.  0.  0.
  -3.4  -3.0 -12.  0.  0.
  -3.8  .80 -1.5 -10.  0.
   2.5 -1.4 -1.5 -1.5 -13.
Z =
  1.0  0.  0.  0.  0.
  0.  .26  .95  -.14  -.70E-01
  0.  .87E-01  -.24E-01  .43  -.90
  0.  .24E-01  .16  .89  .43
  0.  -.96  .26  .22E-01  -.89E-01

QZHES 78
QZHES 79
QZHES 80
QZHES 81
QZHES 82
QZHES 83
QZHES 84
QZHES 85
QZHES 86
QZHES 87
QZHES 88
QZHES 89
QZHES 90
QZHES 91
QZHES 92
QZHES 93
QZHES 94
QZHES 95
QZHES 96
QZHES 97
QZHES 98
QZHES 99
QZHES100
QZHES101
QZHES102
QZHES103
QZHES104
QZHES105
QZHES106
EISP6685
EISP6686
EISP6687
EISP66
EISP6690
EISP6691
EISP6692
EISP6693
EISP6694
EISP6695
EISP6696
EISP6697
EISP6698
EISP6699
EISP6700
EISP6701
EISP6702
EISP6703
EISP6704
EISP6705
EISP6706
EISP6707
EISP6708
EISP6709
EISP6710
EISP6711
EISP6712
EISP6713
EISP6714
R = 0.0E0

DO 25 I = L, N
   B(I,L) = B(I,L) / S
   R = R + B(I,L)**2
25 CONTINUE

R = SIGN(SQRT(R),B(L,L))
B(L,L) = B(L,L) + R
RHO = R * B(L,L)

DO 50 J = L1, N
   T = 0.0E0
   DO 30 I = L, N
      T = T + B(I,L) * B(I,J)
   30 CONTINUE
   T = -T / RHO

DO 40 I = L, N
   B(I,J) = B(I,J) + T * B(I,L)
40 CONTINUE

50 CONTINUE

DO 80 J = I, N
   T = 0.0E0
   DO 60 I = L, N
      T = T + B(I,L) * A(I,J)
   60 CONTINUE
   T = -T / RHO

DO 70 I = L, N
   A(I,J) = A(I,J) + T * B(I,L)
70 CONTINUE

80 CONTINUE

B(L,L) = -S * R

DO 90 I = L1, N
   B(I,L) = 0.0E0
90 CONTINUE

............ REDUCE A TO UPPER HESSENBERG FORM, WHILE
............ KEEPING B TRIANGULAR ............

IF (N .EQ. 2) GO TO 170
NM2 = N - 2

DO 160 K = 1, NM2
   NK1 = NM1 - K

............ FOR L=N-1 STEP -1 UNTIL K+1 DO -- ............
   DO 150 LB = 1, NK1
      L = N - LB
      L1 = L + 1

   ............ ZERO A(L+1,K) ............
150 CONTINUE
160 CONTINUE
170
S = ABS(A(L,K)) + ABS(A(L1,K))
IF (S .EQ. 0.0E0) GO TO 150
U1 = A(L,K) / S
U2 = A(L1,K) / S
R = SIGN(SQRT(U1*U1+U2*U2),U1)
V1 = -(U1 + R) / R
V2 = -U2 / R
U2 = V2 / V1
C
DO 110 J = K, N
   T = A(L,J) + U2 * A(L1,J)
   A(L,J) = A(L,J) + T * V1
   A(L1,J) = A(L1,J) + T * V2
110 CONTINUE
C
A(L1,K) = 0.0E0
C
DO 120 J = L, N
   T = B(L,J) + U2 * B(L1,J)
   B(L,J) = B(L,J) + T * V1
   B(L1,J) = B(L1,J) + T * V2
120 CONTINUE
C
......... ZERO B(L+1,L) ..........
S = ABS(B(L1,L1)) + ABS(B(L1,L))
IF (S .EQ. 0.0E0) GO TO 150
U1 = B(L1,L1) / S
U2 = B(L1,L) / S
R = SIGN(SQRT(U1*U1+U2*U2),U1)
V1 = -(U1 + R) / R
V2 = -U2 / R
U2 = V2 / V1
C
DO 130 I = 1, L1
   T = B(I,L1) + U2 * B(I,L)
   B(I,L1) = B(I,L1) + T * V1
   B(I,L) = B(I,L) + T * V2
130 CONTINUE
C
B(L1,L) = 0.0E0
C
DO 140 I = 1, N
   T = A(I,L1) + U2 * A(I,L)
   A(I,L1) = A(I,L1) + T * V1
   A(I,L) = A(I,L) + T * V2
140 CONTINUE
C
IF (.NOT. MATZ) GO TO 150
C
DO 145 I = 1, N
   T = Z(I,L1) + U2 * Z(I,L)
   Z(I,L1) = Z(I,L1) + T * V1
   Z(I,L) = Z(I,L) + T * V2
145 CONTINUE
C
150 CONTINUE
C
160 CONTINUE
C
170 RETURN
C** THIS PROGRAM VALID ON FTN4 AND FTN5 **
C ROUTINE NAME - PF261=QZIT
FROM EISPACK

------------------------------------------
_C LATEST REVISION
_C

PURPOSE

- THIS SUBROUTINE ACCEPTS A PAIR OF REAL MATRICES, ONE OF THEM IN UPPER HESSENBEG FORM AND THE OTHER IN UPPER TRIANGULAR FORM. IT REDUCES THE HESSENBEG MATRIX TO QUASI-TRIANGULAR FORM USING ORTHOGONAL TRANSFORMATIONS WHILE MAINTAINING THE TRIANGULAR FORM OF THE OTHER MATRIX. IT IS USUALLY PRECEDED QZHES(PF260) AND FOLLOWED BY QZVAL(PF262) AND, POSSIBLY, QZVEC(PF263).

USAGE

- CALL QZIT(NM,N,A,B,EPSI,MATZ,Z,IERR)

ARGUMENTS

NM

- ON INPUT NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM DIMENSION STATEMENT.

N

- ON INPUT N IS THE ORDER OF THE MATRICES.

A

- ON INPUT A CONTAINS A REAL UPPER HESSIENBERG MATRIX. MUST BE OF DIMENSION NM X N.

ON OUTPUT A HAS BEEN REDUCED TO QUASI-TRIANGULAR FORM. THE ELEMENTS BELOW THE FIRST SUBDIAGONAL ARE STILL ZERO AND NO TWO CONSECUTIVE SUBDIAGONAL ELEMENTS ARE NONZERO.

B

- ON INPUT B CONTAINS A REAL UPPER TRIANGULAR MATRIX. MUST BE OF DIMENSION NM X N.

ON OUTPUT B IS STILL IN UPPER TRIANGULAR FORM, ALTHOUGH ITS ELEMENTS HAVE BEEN ALTERED. THE LOCATION B(N,1) IS USED TO STORE EPS1 TIMES THE NORM OF B FOR LATER USE BY QZVAL QZVAL(PF262) AND QZVEC(PF263).

EPSI

- ON INPUT EPSI IS A TOLERANCE USED TO DETERMINE NEGIGLIBLE ELEMENTS. EPSI = 0.0 (OR NEGATIVE) MAY BE INPUT, IN WHICH CASE AN ELEMENT WILL BE NEGLECTED ONLY IF IT IS LESS THAN ROUNDOFF ERROR TIMES THE NORM OF ITS MATRIX. IF THE INPUT EPSI IS POSITIVE, THEN AN ELEMENT WILL BE CONSIDERED NEGIGLIBLE IF IT IS LESS THAN EPSI TIMES THE NORM OF ITS MATRIX. A POSITIVE VALUE OF EPSI MAY RESULT IN FASTER EXECUTION, BUT LESS ACCURATE RESULTS.
MATZ - ON INPUT MATZ SHOULD BE SET TO .TRUE. IF THE RIGHT HAND TRANSFORMATIONS ARE TO BE ACCUMULATED FOR LATER USE IN COMPUTING EIGENVECTORS, AND TO .FALSE. OTHERWISE.

Z - ON INPUT Z CONTAINS, IF MATZ HAS BEEN SET TO .TRUE., THE TRANSFORMATION MATRIX PRODUCED IN QZHES(PF260), IF PERFORMED, OR ELSE THE IDENTITY MATRIX. IF MATZ HAS BEEN SET TO .FALSE., Z IS NOT REFERENCED. MUST BE OF DIMENSION NM X N.

ON OUTPUT Z CONTAINS THE PRODUCT OF THE RIGHT HAND TRANSFORMATIONS (FOR BOTH STEPS) IF MATZ HAS BEEN SET TO .TRUE..

IERR - ON OUTPUT IERR IS SET TO ZERO FOR NORMAL RETURN. J IF THE LIMIT OF 30*N ITERATIONS IS EXHAUSTED WHILE THE J-TH EIGENVALUE IS BEING SOUGHT.

REQUIRED ROUTINES - HC318=EPSLON

REMARKS 1. THIS SUBROUTINE IS THE SECOND STEP OF THE QZ ALGORITHM FOR SOLVING GENERALIZED MATRIX EIGENVALUE PROBLEMS, SIAM J. NUMER. ANAL. 10, 241-256(1973) BY MOLER AND STEWART, AS MODIFIED IN TECHNICAL NOTE NASA TN D-7305(1973) BY WARD.

EXAMPLE :

PROGRAM TQZIT(OUTPUT,TAPE6=OUTPUT)
DIMENSION A(5,5),B(5,5),Z(5,5)
LOGICAL MATZ
N = 5
NM = 5
MATZ = .TRUE.
EPS1 = 0.0E0

DATA A /10.,2.,3.,2*1.,2.,12.,1.,2.,1.,3.,1.,11.,
* 1.,1.,2.,1.,9.,3*1.,1.,1.,15. /
DATA B /12.,1.,-1.,2.,2*1.,14.,1.,-1.,1.,-1.,1.,
* 16.,-1.,1.,2.,-1.,-1.,-1.,12.,-1.,3*1.,-1.,11. /

CALL QZHES(NM,N,A,B,MATZ,Z)
CALL QZIT(NM,N,A,B,EPS1,MATZ,Z,IERR)
WRITE(6,99) IERR
FORMAT(IHI,8H IERR = ,I4)
WRITE(6,100) ((A(I,J),I=1,5),J=1,5),((B(I,J),I=1,5),J=1,5),
* ((Z(I,J),I=1,5),J=1,5)
FORMAT(1H ,5H A = /5(1H ,5(G8.2,2X))
* 5H B = /5(1H ,5(G8.2,2X))
* 5H Z = /5(1H ,5(G8.2,2X))
STOP
END
C OUTPUT :
C     IERR = 0
A =
   -1.5  -1.3  0.0  0.0  0.0
    7.4  0.0  0.0  0.0
   -1.5 -16.0  0.0  0.0
   -2.2  0.96 -10.0  0.0
   -2.6 -0.31  1.2  1.7  -8.6
B =
  -9.9  0.0  0.0  0.0  0.31E-12
   -2.9  17.0  0.0  0.0
   -1.3 -2.1 -14.0  0.0
   -1.9  1.7  96.0 -11.0
   -2.6 -3.2  1.3  2.1 -13.0
Z =
  0.28 -0.32E-01  0.16  0.24 -0.91
  0.52 -0.24  0.66  0.48 -0.64E-01
  0.49 -0.56  0.49  0.45 -0.75E-01
  0.49 -0.48 -0.29  0.44 -0.38
   -0.25 -0.63  0.45  0.57 -0.94E-01

--------
SUBROUTINE QZIT(NM,N,A,B,EPSI,MATZ,Z,IERR)

IMPLICIT REAL*8 (A-H,O-Z)
INTEGER I,J,K,L,N,EN,KI,K2,LD,LL,LI,NA,NM,ISH,ITN,ITS,K1,L1,
      ENM2,IERR,LOL1,ENORN
REAL*8 A(NM,N),B(NM,N),Z(NM,N)
REAL*8 R,S,T,AI,A2,A3,EP,SH,UI,U2,U3,V1,V2,V3,ANI,A11,
      V4,A12,A21,A22,A33,A43,A44,B11,B12,B22,B33,B34,
X     B44,EPSA,EPSB,EPS1,ANORM,BNORM,EPSLON
LOGICAL MATZ,NOTLAS
IERR = 0

....... COMPUTE EPSA,EPSB .........
ANORM = 0.0E0
BNORM = 0.0E0

DO 30 I = 1, N
   ANI = 0.0E0
   IF (I .NE. 1) ANI = ABS(A(I,1-I))
      BNI = 0.0E0
   30 CONTINUE

DO 20 J = 1, N
   ANI = ANI + ABS(A(I,J))
   BNI = BNI + ABS(B(I,J))
   20 CONTINUE

IF (ANI .GT. ANORM) ANORM = ANI
IF (BNI .GT. BNORM) BNORM = BNI

30 CONTINUE

IF (ANORM .EQ. 0.0E0) ANORM = 1.0E0
IF (BNORM .EQ. 0.0E0) BNORM = 1.0E0
EP = EPS1
IF (EP .GT. 0.0E0) GO TO 50
      USE ROUNDOFF LEVEL IF EPS1 IS ZERO .........
      EP = EPSLON(1.0E0)
      50 EPSA = EP * ANORM
      EPSB = EP * BNORM

QZIT 121
QZIT 122
QZIT 123
QZIT 124
QZIT 125
QZIT 126
QZIT 127
QZIT 128
QZIT 129
QZIT 130
QZIT 131
QZIT 132
QZIT 133
QZIT 134
QZIT 135
QZIT 136
QZIT 137
QZIT 138
QZIT 139
QZIT 140
QZIT 141
QZIT 142
QZIT 143

EISPN6837
EISPN6838
EISPN6839
EISPN6840
EISPN6841
EISPN6842
EISPN6843
EISPN6844
EISPN6845
EISPN6846
EISPN6847
EISPN6848
EISPN6849
EISPN6850
EISPN6851
EISPN6852
EISPN6853
EISPN6854
EISPN6855
EISPN6856
EISPN6857
EISPN6858
EISPN6859
EISPN6860
EISPN6861
EISPN6862
EISPN6863
EISPN6864
EISPN6865
EISPN6866
EISPN6867
EISPN6868
EISPN6869
EISPN6870
EISPN6871
C REDUCE A TO QUASI-TRIANGULAR FORM, WHILE
C KEEPING B TRIANGULAR ............
LOR1 = 1
ENORN = N
EN = N
ITN = 30*N
BEGIN QZ STEP ............
60 IF (EN .LE. 2) GO TO 1001
IF (.NOT. MATZ) ENORN = EN
ITS = 0
NA = EN - 1
ENM2 = NA - 1
70 ISH = 2
C CHECK FOR CONVERGENCE OR REDUCIBILITY.
C FOR L=EN STEP -1 UNTIL 1 DO -- .........
DO 80 LL = 1, EN
   LM1 = EN - LL
   L = LM1 + 1
   IF (L .EQ. 1) GO TO 95
   IF (ABS(A(L,LM1)) .LE. EPSA) GO TO 90
80 CONTINUE
C A(L,LM1) = 0.0E0
   IF (L .LT. NA) GO TO 95
C 1-BY-1 OR 2-BY-2 BLOCK ISOLATED ............
   EN = LM1
   GO TO 60
C CHECK FOR SMALL TOP OF B ............
95 LD = L
100 L1 = L + 1
   B11 = B(L,L)
   IF (ABS(B11) .GT. EPSB) GO TO 120
   B(L,L) = 0.0E0
   S = ABS(A(L,L)) + ABS(A(L1,L))
   U1 = A(L,L) / S
   U2 = A(L1,L) / S
   R = SIGN(SQRT(U1*U1+U2*U2),U1)
   V1 = -(U1 + R) / R
   V2 = -U2 / R
   U2 = V2 / V1
   DO 110 J = L, ENORN
      T = A(L,J) + U2 * A(L1,J)
      A(L,J) = A(L,J) + T * V1
      A(L1,J) = A(L1,J) + T * V2
      T = B(L,J) + U2 * B(L1,J)
      B(L,J) = B(L,J) + T * V1
      B(L1,J) = B(L1,J) + T * V2
110 CONTINUE
C A(L,LM1) = -A(L,LM1)
LM1 = L
L = L1
GO TO 90
120 A11 = A(L,L) / B11
   A21 = A(L1,L) / B11
   IF (ISH .EQ. 1) GO TO 140
C ITERATION STRATEGY ............
   IF (ITN .EQ. 0) GO TO 1000
   IF (ITS .EQ. 10) GO TO 155
C DETERMINE TYPE OF SHIFT
B22 = B(L1, L1)
IF (ABS(B22) .LT. EPSB) B22 = EPSB
B33 = B(NA, NA)
IF (ABS(B33) .LT. EPSB) B33 = EPSB
B44 = B(EN, EN)
IF (ABS(B44) .LT. EPSB) B44 = EPSB
A33 = A(NA, NA) / B33
A34 = A(NA, EN) / B44
A43 = A(EN, NA) / B33
A44 = A(EN, EN) / B44
T = 0.5E0 * (A43 * B34 - A34 - A33 * A44)
R = T * T + A34 * A43 - A33 * A44
IF (R .LT. 0.0E0) GO TO 150

C DETERMINE SINGLE SHIFT ZEROTH COLUMN OF A
ISH = 1
R = SQRT(R)
SH = -T + R
S = -T - R
IF (ABS(S-A44) .LT. ABS(SH-A44)) SH = S
C LOOK FOR TWO CONSECUTIVE SMALL SUB-DIAGONAL ELEMENTS OF A.
   FOR L=EN-2 STEP -1 UNTIL LD DO --
   DO 130 LL = LD, ENM2
   L = ENM2 + LD - LL
   IF (L .EQ. LD) GO TO 140
   LM1 = L - 1
   L1 = L + 1
   T = A(L, L)
   IF (ABS(B(L, L)) .GT. EPSB) T = T - SH * B(L, L)
   IF (ABS(A(L, LM1)) .LE. ABS(T/A(L1, L)) * EPSA) GO TO 100
130 CONTINUE

C A1 = A11 - SH
  A2 = A21
  IF (L .NE. LD) A(L, LM1) = -A(L, LM1)
  GO TO 160
C DETERMINE DOUBLE SHIFT ZEROTH COLUMN OF A
150 A12 = A(L, L1) / B22
  A22 = A(L1, L1) / B22
  B12 = B(L, L1) / B22
  / A21 + A12 - A11 * B12
  A2 = (A22 - A11) - A21 * B12 - (A33 - A11) - (A44 - A11)
  + A43 * B34
  A3 = A(L1+1, L1) / B22
  GO TO 160
C AD HOC SHIFT
155 A1 = 0.0E0
  A2 = 1.0E0
  A3 = 1.1605E0
160 ITS = ITS + 1
  ITN = ITN - 1
  IF (.NOT. MATZ) LOR1 = LD
C MAIN LOOP
160 K = L, NA
  NOTLAS = K .NE. NA .AND. ISH .EQ. 2
  K1 = K + 1
  K2 = K + 2
```
C

KM1 = MAX0(K-1,L)
LL = MIN0(EN,K1+ISH)
IF (NOTLAS) GO TO 190
C

............. ZERO A(K+1,K-1) ..........  
IF (K .EQ. L) GO TO 170
A1 = A(K,KM1)
A2 = A(K1,KM1)
C

170
S = ABS(A1) + ABS(A2)
IF (S .EQ. 0.0E0) GO TO 70
U1 = A1 / S
U2 = A2 / S
R = SIGN(SQRT(U1*U1+U2*U2),U1)
V1 = -(U1 + R) / R
V2 = -U2 / R
U2 = V2 / V1
C

DO 180 J = KM1, ENORN
T = A(K,J) + U2 * A(K1,J)
A(K,J) = A(K,J) + T * V1
A(K1,J) = A(K1,J) + T * V2
C

T = B(K,J) + U2 * B(K1,J)
B(K,J) = B(K,J) + T * V1
B(K1,J) = B(K1,J) + T * V2
C

180 CONTINUE
C

190
C

190
C

CONTINUE
C

(IF (K .NE. L) A(K1,KM1) = 0.0E0
GO TO 240
C

200
S = ABS(A1) + ABS(A2) + ABS(A3)
IF (S .EQ. 0.0E0) GO TO 260
C

U1 = A1 / S
U2 = A2 / S
U3 = A3 / S
C

R = SIGN(SQRT(U1*U1+U2*U2+U3*U3),U1)
V1 = -(U1 + R) / R
V2 = -U2 / R
V3 = -U3 / R
U2 = V2 / V1
U3 = V3 / V1
C

DO 210 J = KM1, ENORN
T = A(K,J) + U2 * A(K1,J) + U3 * A(K2,J)
A(K,J) = A(K,J) + T * V1
A(K1,J) = A(K1,J) + T * V2
A(K2,J) = A(K2,J) + T * V3
C

T = B(K,J) + U2 * B(K1,J) + U3 * B(K2,J)
B(K,J) = B(K,J) + T * V1
B(K1,J) = B(K1,J) + T * V2
B(K2,J) = B(K2,J) + T * V3
C

210 CONTINUE
C

(IF (K .EQ. L) GO TO 220
A(K1,KM1) = 0.0E0
A(K2,KM1) = 0.0E0
C

220
S = ABS(B(K2,K2)) + ABS(B(K2,K1)) + ABS(B(K2,K))
C
```
IF (S .EQ. 0.0E0) GO TO 240
U1 = B(K2,K2) / S
U2 = B(K2,K1) / S
U3 = B(K2,K) / S
R = SIGN(SQRT(U1*U1+U2*U2+U3*U3),U1)
V1 = -(U1 + R) / R
V2 = -U2 / R
V3 = -U3 / R
U2 = V2 / V1
U3 = V3 / V1
C
DO 230 I = LOR1, LL
   T = A(I,K2) + U2 * A(I,K1) + U3 * A(I,K)
   A(I,K2) = A(I,K2) + T * V1
   A(I,K1) = A(I,K1) + T * V2
   A(I,K) = A(I,K) + T * V3
   T = B(I,K2) + U2 * B(I,K1) + U3 * B(I,K)
   B(I,K2) = B(I,K2) + T * V1
   B(I,K1) = B(I,K1) + T * V2
   B(I,K) = B(I,K) + T * V3
230 CONTINUE
C
B(K2,K) = 0.0E0
B(K2,K1) = 0.0E0
IF (.NOT. MATZ) GO TO 240
C
DO 235 I = 1, N
   T = Z(I,K2) + U2 * Z(I,K1) + U3 * Z(I,K)
   Z(I,K2) = Z(I,K2) + T * V1
   Z(I,K1) = Z(I,K1) + T * V2
   Z(I,K) = Z(I,K) + T * V3
235 CONTINUE
C
............ ZERO B(K+I,K) ................
C
DO 240 I = LOR1, LL
   T = A(I,K1) + U2 * A(I,K)
   A(I,K1) = A(I,K1) + T * V1
   A(I,K) = A(I,K) + T * V2
   T = B(I,K1) + U2 * B(I,K)
   B(I,K1) = B(I,K1) + T * V1
   B(I,K) = B(I,K) + T * V2
240 CONTINUE
-- 260 CONTINUE C ........... END QZ STEP ...........
GO TO 70
-C ........... SET ERROR -- ALL EIGENVALUES HAVE NOT C CONVERGED AFTER 30*N ITERATIONS ............
1000 IERR = EN
-C ........... SAVE EPSB FOR USE BY QZVAL AND QZVEC ............
1001 IF (N .GT. 1) B(N,1) = EPSB RETURN
-C** THIS PROGRAM VALID ON FTN4 AND FTN5 **
END

ROUTINE NAME - PF262=QZVAL
FROM EISPACK

LATEST REVISION - AUGUST 1,1984
COMPUTER SCIENCES CORP., HAMPTON, VA.

PURPOSE - THIS SUBROUTINE ACCEPTS A PAIR OF REAL MATRICES, ONE OF THEM IN QUASI-TRIANGULAR FORM AND THE OTHER IN UPPER TRIANGULAR FORM. IT REDUCES THE QUASI-TRIANGULAR MATRIX FURTHER, SO THAT ANY REMAINING 2-BY-2 BLOCKS CORRESPOND TO PAIRS OF COMPLEX EIGENVALUES, AND RETURNS QUANTITIES WHOSE RATIOS GIVE THE GENERALIZED EIGENVALUES. IT IS USUALLY PRECEDED BY QZHES(PF260) AND QZIT(PF261) AND MAY BE FOLLOWED BY QZVEC(PF263).

USAGE - CALL QZVAL(NM,N,A,B,ALFR,ALFI,BETA,MATZ,Z)

ARGUMENTS NM - ON INPUT NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM DIMENSION STATEMENT.

N - ON INPUT N IS THE ORDER OF THE MATRICES.

A - ON INPUT A CONTAINS A REAL UPPER QUASI-TRIANGULAR MATRIX. MUST BE OF DIMENSION NM X N.
ON OUTPUT A HAS BEEN REDUCED FURTHER TO A QUASI-TRIANGULAR MATRIX IN WHICH ALL NONZERO SUBDIAGONAL ELEMENTS CORRESPOND TO PAIRS OF COMPLEX EIGENVALUES.

B - ON INPUT B CONTAINS A REAL UPPER TRIANGULAR MATRIX. MUST BE OF DIMENSION NM X N.
IN ADDITION, LOCATION B(N,1) CONTAINS THE TOLERANCE QUANTITY (EPSB) COMPUTED AND SAVED IN QZIT(PF261).
ON OUTPUT B IS STILL IN UPPER TRIANGULAR FORM, ALTHOUGH ITS ELEMENTS HAVE BEEN ALTERED. B(N,1) IS UNALTEDER.
ALFR - ON OUTPUT ALFR CONTAINS THE REAL PART OF THE
DIAGONAL ELEMENTS OF THE TRIANGULAR MATRIX
THAT WOULD BE OBTAINED IF A WERE REDUCED
COMPLETELY TO TRIANGULAR FORM BY UNITARY
TRANSFORMATIONS. NON-ZERO VALUES OF ALFI
OCCUR IN PAIRS, THE FIRST MEMBER POSITIVE AND
THE SECOND NEGATIVE.  MUST BE OF DIMENSION N.

ALFI - ON OUTPUT ALFI CONTAINS THE IMAGINARY PART
OF THE DIAGONAL ELEMENTS OF THE TRIANGULAR
MATRIX THAT WOULD BE OBTAINED IF A WERE
REDUCED COMPLETELY TO TRIANGULAR FORM BY
UNITARY TRANSFORMATIONS. NON-ZERO VALUES
OF ALFI OCCUR IN PAIRS, THE FIRST MEMBER
POSITIVE AND THE SECOND NEGATIVE.  MUST BE OF DIMENSION N.

BETA - ON OUTPUT BETA CONTAINS THE DIAGONAL ELEMENTS
OF THE CORRESPONDING B, NORMALIZED TO BE REAL
AND NON-NEGATIVE. THE GENERALIZED EIGENVALUES
ARE THEN THE RATIOS ((ALFR+I*ALFI)/BETA).  MUST BE OF DIMENSION N.

MATZ - ON INPUT MATZ SHOULD BE SET TO .TRUE. IF
THE RIGHT HAND TRANSFORMATIONS ARE TO BE
ACCUMULATED FOR LATER USE IN COMPUTING
EIGENVECTORS, AND TO .FALSE. OTHERWISE.

Z - ON INPUT Z CONTAINS, IF MATZ HAS BEEN SET
TO .TRUE., THE TRANSFORMATION MATRIX PRODUCED
IN THE REDUCTIONS BY QZHS(PF260) AND QZIT
(PF261) IF PERFORMED, OR ELSE THE IDENTITY
MATRIX. IF MATZ HAS BEEN SET TO .FALSE., Z
IS NOT REFERENCED.  MUST BE OF DIMENSION NM X N.

ON OUTPUT Z CONTAINS THE PRODUCT OF THE
RIGHT HAND TRANSFORMATIONS (FOR ALL THREE
STEPS) IF MATZ HAS BEEN SET TO .TRUE.

REQUIRED ROUTINES - NONE

REMARKS 1. THIS SUBROUTINE IS THE THIRD STEP OF THE QZ
ALGORITHM FOR SOLVING GENERALIZED MATRIX
EIGENVALUE PROBLEMS, SIAM J. NUMER. ANAL. 10,
241-256(1973) BY MOLER AND STEWART.

EXAMPLE :
PROGRAM TQZVAL(OUTPUT,TAPE6=OUTPUT)
DIMENSION A(5,5),B(5,5),ALFR(5),ALFI(5),BETA(5),Z(5,5)
LOGICAL MATZ

N = 5
NM = 5
MATZ = .TRUE.
EPS1 = 0.0E0
DATA A /10.,2.,3.,2*1.,2.,12.,1.,1.,2.,1.,9.,3*1.,-1.,1.,15. /
DATA B /12.,1.,-1.,2.,2*1.,14.,1.,-1.,1.,-1.,1. /

CALL QZHES(NM,N,A,B,MATZ,Z)
CALL QZIT(NM,N,A,B,EPSI,MATZ,Z,IERR)
WRITE(6,99) IERR
WRITE(6,100) ALFR,ALFI,BETA,((Z(I,J),I=I,5),J=I,5)

FORMAT(IHI,8H IERR = ,I4)
FORMAT(IH ,8H ALFR =
,5(G8.2,2X)/
8H ALFI =
,5(G8.2,2X)/
8H BETA =
,5(G8.2,2X)/
5H Z =/5(1H ,5(G8.2,2X)/))

STOP
END

OUTPUT :
IERR = 0
ALFR =
15. 7.2 16. 10. 8.6
ALFI =
0. 0. 0. 0. 0.
BETA =
Z =
.24 -.54E-01 .21 -.27 -.91
-.54 .25 .65 -.46 .13
.49 .56 .49 .45 .75E-01
-.60 .48 -.29 .44 -.38
-.25 -.63 .45 .57 -.94E-01

SUBROUTINE QZVAL(NM,N,A,B,ALFR,ALFI,BETA,MATZ,Z)

IMPLICIT REAL*8 (A-H,O-Z)
INTEGER I,J,N,EN,NA,NM,NN,ISW
REAL*8 A(NM,N),B(NM,N),ALFR(N),ALFI(N),BETA(N),Z(NM,N)
REAL*8 C,D,E,R,S,T,AN,A1,A2,BN,CQ,CZ,DI,DR,EL,FI,UI,
X U2,V1,V2,A11,A12,A21,A22,B11,B12,B22,QI,SQI,SQR,
X SSI,SSR,SZI,SZR,AIII,AIIR,AI2I,AI2R,A22I,A22R,EPSB
LOGICAL MATZ
EPSB = B(N,I)
ISW = 1

......... FIND EIGENVALUES OF QUASI-TRIANGULAR MATRICES.
FOR EN=N STEP -1 UNTIL 1 DO -- ..........

DO 510 NN = I, N

EN = N + 1 - NN
NA = EN - 1
IF (ISW .EQ. 2) GO TO 505
IF (EN .EQ. 1) GO TO 410
IF (A(EN,NA) .NE. 0.0E0) GO TO 420

510 ALFR(EN) = A(EN,EN)
 IF (B(EN,EN) .LT. 0.0E0) ALFR(EN) = -ALFR(EN)
 BETA(EN) = ABS(B(EN,EN))
ALFI(EN) = 0.0E0
GO TO 510

C ............ 2-BY-2 BLOCK ............

420 IF (ABS(B(NA,NA)) .LE. EPSB) GO TO 455
IF (ABS(B(EN,EN)) .GT. EPSB) GO TO 430
A1 = A(EN,EN)
A2 = A(EN,NA)
BN = 0.0E0
GO TO 435

430 AN = ABS(A(NA,NA)) + ABS(A(NA,EN)) + ABS(A(EN,NA))
+ ABS(A(EN,EN))
BN = ABS(B(NA,NA)) + ABS(B(NA,EN)) + ABS(B(EN,EN))
A11 = A(NA,NA) / AN
A12 = A(NA,EN) / AN
A21 = A(EN,NA) / AN
A22 = A(EN,EN) / AN
B11 = B(NA,NA) / BN
B12 = B(NA,EN) / BN
B22 = B(EN,EN) / BN
E = A11 / B11
EI = A22 / B22
S = A21 / (B11 * B22)
T = (A22 - E * B22) / B22
IF (ABS(E) .LE. ABS(EI)) GO TO 431
E = EI
T = (A11 - E * B11) / B11

431 C = 0.5E0 * (T - S * B12)
D = C * C + S * (A12 - E * B12)
IF (D .LT. 0.0E0) GO TO 480
C ............ TWO REAL ROOTS. ............

432 A1 = A22
A2 = A21
GO TO 435

435 S = ABS(A1) + ABS(A2)
U1 = A1 / S
U2 = A2 / S
R = SIGN(SQRT(U1*U1+U2*U2),U1)
V1 = -(U1 + R) / R
V2 = -U2 / R
U2 = V2 / V1

C ............ CHOOSE AND APPLY REAL Z ............

DO 440 I = 1, EN
T = A(I,EN) + U2 * A(I,NA)
A(I,EN) = A(I,EN) + T * V1
A(I,NA) = A(I,NA) + T * V2
T = B(I,EN) + U2 * B(I,NA)
B(I,EN) = B(I,EN) + T * V1
B(I,NA) = B(I,NA) + T * V2
CONTINUE

C

CONTINUE
```plaintext
IF (.NOT. MATZ) GO TO 450

DO 445 I = 1, N
   T = Z(I,EN) + U2 * Z(I,NA)
   Z(I,EN) = Z(I,EN) + T * V1
   Z(I,NA) = Z(I,NA) + T * V2
CONTINUE

445

IF (BN .EQ. 0.0E0) GO TO 475
IF (AN .LT. ABS(E) * BN) GO TO 455
   A1 = B(NA,NA)
   A2 = B(EN,NA)
   GO TO 460

455
   A1 = A(NA,NA)
   A2 = A(EN,NA)

C .......... CHOOSE AND APPLY REAL Q ..........

460
   S = ABS(A1) + ABS(A2)
   IF (S .EQ. 0.0E0) GO TO 475
   U1 = A1 / S
   U2 = A2 / S
   R = SIGN(SQRT(U1*U1+U2*U2),U1)
   V1 = -(U1 + R) / R
   V2 = -U2 / R
   U2 = V2 / V1

C

DO 470 J = NA, N
   T = A(NA,J) + U2 * A(EN,J)
   A(NA,J) = A(NA,J) + T * V1
   A(EN,J) = A(EN,J) + T * V2
   T = B(NA,J) + U2 * B(EN,J)
   B(NA,J) = B(NA,J) + T * V1
   B(EN,J) = B(EN,J) + T * V2
CONTINUE

C

475
   A(EN,NA) = 0.0E0
   B(EN,NA) = 0.0E0
   ALFR(NA) = A(NA,NA)
   ALFR(EN) = A(EN,EN)
   IF (B(NA,NA) .LT. 0.0E0) ALFR(NA) = -ALFR(NA)
   IF (B(EN,EN) .LT. 0.0E0) ALFR(EN) = -ALFR(EN)
   BETA(NA) = ABS(B(NA,NA))
   BETA(EN) = ABS(B(EN,EN))
   ALFI(EN) = 0.0E0
   ALFI(NA) = 0.0E0
   GO TO 505

C .......... TWO COMPLEX ROOTS ..........

480
   E = E + C
   EI = SQRT(-D)
   A11R = A11 - E * B11
   A11I = EI * B11
   A12R = A12 - E * B12
   A12I = EI * B12
   A22R = A22 - E * B22
   A22I = EI * B22
   IF (ABS(A11R) + ABS(A11I) + ABS(A12R) + ABS(A12I) .LT.
      X
      ABS(A21) + ABS(A22R) + ABS(A22I)) GO TO 482
   A1 = A12R
   A1I = A12I
   A2 = -A11R
   A2I = -A11I
```

GO TO 485

482 A1 = A22R
A1I = A22I
A2 = -A21
A2I = 0.0E0

C ............ CHOOSE COMPLEX Z ............

485 CZ = SQRT(A1*A1+A1I*A1I)

IF (CZ .EQ. 0.0E0) GO TO 487

SZR = (A1 * A2 + A1I * A2I) / CZ
S2I = (A1 * A2I - A1I * A2) / CZ
R = SQRT(CZ*CZ+SZR*SZR+S2I*S2I)
CZ = CZ / R
SZR = SZR / R
S2I = S2I / R
GO TO 490

487 SZR = 1.0E0
S2I = 0.0E0

490 IF (AN .LT. (ABS(E) + EI) * BN) GO TO 492

A1 = CZ * B11 + SZR * B12
A1I = S2I * B12
A2 = SZR * B22
A2I = S2I * B22
GO TO 495

492 A1 = CZ * A11 + SZR * A12
A1I = S2I * A12
A2 = CZ * A21 + SZR * A22
A2I = S2I * A22

C ............ CHOOSE COMPLEX Q ............

495 CQ = SQRT(A1*A1+A1I*A1I)

IF (CQ .EQ. 0.0E0) GO TO 497

SQR = (A1 * A2 + A1I * A2I) / CQ
SQI = (A1 * A2I - A1I * A2) / CQ
R = SQRT(CQ*CQ+SQR*SQR+SQI*SQI)
CQ = CQ / R
SQR = SQR / R
SQI = SQI / R
GO TO 500

497 SQR = 1.0E0
SQI = 0.0E0

C ............ COMPUTE DIAGONAL ELEMENTS THAT WOULD RESULT
IF TRANSFORMATIONS WERE APPLIED ............

500 SSR = SQR * SZR + SQI * S2I
SSI = SQR * S2I - SQI * SZR
I = 1

TR = CQ * CZ * A11 + CQ * SZR * A12 + SQR * CZ * A21
+ SQR * A22
TI = CQ * S2I * A12 - SQI * CZ * A21 + SSI * A22
DR = CQ * CZ * B11 + CQ * SZR * B12 + SSR * B22
DI = CQ * S2I * B12 + SSI * B22
GO TO 503

502 I = 2

TR = SSR * A11 - SQR * CZ * A12 - CQ * SZR * A21
+ CQ * CZ * A22
TI = -SSI * A11 - SQI * CZ * A12 + CQ * S2I * A21
DR = SSR * B11 - SQR * CZ * B12 + CQ * CZ * B22
DI = -SSI * B11 - SQI * CZ * B12

503 T = TI * DR - TR * DI
J = NA
IF (T .LT. 0.0E0) J = EN
R = SQRT(DR*DR+DI*DI)
\[
\begin{align*}
\text{BETA(J)} &= BN \times R \\
\text{ALFR(J)} &= AN \times \left( TR \times DR + TI \times DI \right) / R \\
\text{ALFI(J)} &= AN \times T / R \\
\text{IF (I .EQ. 1) GO TO 502} \\
505 \text{ CONTINUE} \\
510 \text{ B(N,1) = EPSB} \\
\end{align*}
\]

RETURN

** THIS PROGRAM VALID ON FTN4 AND FTN5 **

END

ROUTINE NAME - PF263=QZVEC

FROM EISPACK

LATEST REVISION - AUGUST 1, 1984
COMPUTER SCIENCES CORP., HAMPTON, VA.

PURPOSE - THIS SUBROUTINE ACCEPTS A PAIR OF REAL MATRICES, ONE OF THEM IN QUASI-TRIANGULAR FORM (IN WHICH EACH 2-BY-2 BLOCK CORRESPONDS TO A PAIR OF COMPLEX EIGENVALUES) AND THE OTHER IN UPPER TRIANGULAR FORM. IT COMPUTES THE EIGENVECTORS OF THE TRIANGULAR PROBLEM AND TRANSFORMS THE RESULTS BACK TO THE ORIGINAL COORDINATE SYSTEM. IT IS USUALLY PRECEDED BY QZHES(PF260), QZIT(PF261), AND QZVAL(PF262).

USAGE - CALL QZVEC(NM,N,A,B,ALFR,ALFI,BETA,Z)

ARGUMENTS

NM - ON INPUT NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM DIMENSION STATEMENT.

N - ON INPUT N IS THE ORDER OF THE MATRICES.

A - ON INPUT A CONTAINS A REAL UPPER QUASI-TRIANGULAR MATRIX. MUST BE OF DIMENSION NM X N.

- ON OUTPUT A IS UNALTERED. ITS SUBDIAGONAL ELEMENTS PROVIDE INFORMATION ABOUT THE STORAGE OF THE COMPLEX EIGENVECTORS.

B - ON INPUT B CONTAINS A REAL UPPER TRIANGULAR MATRIX. IN ADDITION, LOCATION B(N,1) CONTAINS THE TOLERANCE QUANTITY (EPSB) COMPUTED AND SAVED IN QZIT(PF261). MUST BE OF DIMENSION NM X N.

- ON OUTPUT B HAS BEEN DESTROYED.

ALFR - ON INPUT ALFR IS A VECTOR SUCH THAT THE RATIOS ((ALFR+I*ALFI)/BETA) ARE THE GENERALIZED EIGENVALUES. THEY ARE USUALLY
ALFI - ON INPUT ALFI IS A VECTOR SUCH THAT THE RATIOS
((ALFR+I*ALFI)/BETA) ARE THE GENERALIZED
EIGENVALUES. THEY ARE USUALLY OBTAINED FROM
QZVAL(PF262).
MUST BE OF DIMENSION N.

BETA - ON INPUT BETA IS A VECTOR SUCH THAT THE RATIOS
((ALFR+I*ALFI)/BETA) ARE THE GENERALIZED
EIGENVALUES. THEY ARE USUALLY OBTAINED FROM
QZVAL(PF262).
MUST BE OF DIMENSION N.

Z - ON INPUT Z CONTAINS THE TRANSFORMATION MATRIX
PRODUCED IN THE REDUCTIONS BY QZHE(PF260),
QZIT(PF261), AND QZVAL(PF262), IF PERFORMED. IF THE EIGENVECTORS OF THE TRIANGULAR PROBLEM
ARE DESIRED, Z MUST CONTAIN THE IDENTITY
MATRIX.
MUST BE OF DIMENSION NM X N.

ON OUTPUT Z CONTAINS THE REAL AND IMAGINARY
PARTS OF THE EIGENVECTORS. IF ALFI(I) .EQ.
0.0, THE I-TH EIGENVALUE IS REAL AND THE I-TH
COLUMN OF Z CONTAINS ITS EIGENVECTOR. IF
ALFI(I) .NE. 0.0, THE I-TH EIGENVALUE IS
COMPLEX. IF ALFI(I) .GT. 0.0, THE EIGENVALUE IS THE FIRST OF A COMPLEX PAIR AND THE I-TH
AND (I+1)-TH COLUMNS OF Z CONTAIN ITS EIGEN-
VECTOR. IF ALFI(I) .LT. 0.0, THE EIGEN-
VALUE IS THE SECOND OF A COMPLEX PAIR AND THE
(I-1)-TH AND I-TH COLUMNS OF Z CONTAIN THE
CONJUGATE OF ITS EIGENVECTOR. EACH EIGEN-
VECTOR IS NORMALIZED SO THAT THE MODULUS
OF ITS LARGEST COMPONENT IS 1.0.

REQUIRED ROUTINES - NONE

REMARKS 1. THIS SUBROUTINE IS THE OPTIONAL FOURTH STEP
OF THE QZ ALGORITHM FOR SOLVING GENERALIZED
MATRIX EIGENVALUE PROBLEMS, SIAM J. NUMER.
ANAL. 10, 241-256(1973) BY MOLER AND STEWART.

EXAMPLE:
PROGRAM TQZVEC(OUTPUT,TAPE6=OUTPUT)
DIMENSION A(5,5),B(5,5),ALFR(5),ALFI(5),BETA(5),Z(5,5)
LOGICAL MATZ

N = 5
NM = 5
MATZ = .TRUE.
EPS1 = 0.0E0

DATA A /10.,2.,3.,2*1.,2.,12.,1.,2.,1.,3.,1.,11.,
  1.,-1.,1.,2.,1.,9.,3*1.,-1.,1.,15.
DATA B /12.,1.,-1.,2.,2*1.,14.,1.,-1.,1.,-1.,1.,
SUBROUTINE QZVEC(NM,N,A,B,ALFR,ALFI,BETA,Z)

implicit real*8 (a-h,o-z)
INTEGER I,J,K,M,N,EN,II,JJ,NA,NM,NN,ISW,ENM2
REAL*8 A(NM,N),B(NM,N),ALFR(N),ALFI(N),BETA(N),Z(NM,N)
REAL*8 D,Q,R,S,T,W,X,Y,DI,DR,RA,RR,SA,TI,TR,TI,T2,WI,XZ,ZI,ALFM,ALMI,ALMR,BETM,EPSB
EPSB = B(N,1)
ISW = 1

............... FOR EN=N STEP -1 UNTIL 1 DO -- ................
DO 800 NN = i, N
EN = N + 1 - NN
NA = EN - 1
IF (ISW .EQ. 2) GO TO 795
IF (ALFI(EN) .NE. 0.0E0) GO TO 710
............... REAL VECTOR ...........
M = EN
B(EN,EN) = 1.0E0
IF (NA .EQ. 0) GO TO 800
ALFM = ALFR(M)
BETM = BETA(M)
............... FOR I=EN-1 STEP -1 UNTIL 1 DO -- ............
DO 700 II = i, NA
I = EN - II
W = BETM * A(I,I) - ALFM * B(I,I)
R = 0.0E0

C
DO 610 J = M, EN
610 R = R + (BETM * A(I,J) - ALFM * B(I,J)) * B(J,EN)

C
IF (I .EQ. 1) OR. ISW .EQ. 2) GO TO 630
IF (BETM * A(I-1,1) .EQ. 0.0E0) GO TO 630
ZZ = W
S = R
GO TO 690
630 M = I
IF (ISW .EQ. 2) GO TO 640

C  .......... REAL 1-BY-1 BLOCK ..........
T = W
IF (W .EQ. 0.0E0) T = EPSB
B(I,EN) = -R / T
GO TO 700

C  .......... REAL 2-BY-2 BLOCK ..........
640  X = BETM * A(I,I+1) - ALFM * B(I,I+1)
Y = BETM * A(I+1,I)
Q = W * ZZ - X * Y
T = (X * S - ZZ * R) / Q
B(I,EN) = T
IF (ABS(X) .LE. ABS(ZZ)) GO TO 650
B(I+1,EN) = (-R - W * T) / X
GO TO 690

650  B(I+1,EN) = (-S - Y * T) / ZZ
ISW = 3 - ISW
GO TO 700

700  CONTINUE

C  .......... END REAL VECTOR ..........
GO TO 800

-C  .......... COMPLEX VECTOR ..........
710  M = NA
ALMR = ALFR(M)
ALMI = ALFI(M)
BETM = BETA(M)

C  .......... LAST VECTOR COMPONENT CHOSEN IMAGINARY SO THAT
EIGENVECTOR MATRIX IS TRIANGULAR ..........

Y = BETM * A(EN,NA)
B(NA,NA) = -ALMI * B(EN,EN) / Y
B(NA,EN) = (ALMR * B(EN,EN) - BETM * A(EN,EN)) / Y
B(EN,NA) = 0.0E0
B(EN,EN) = 1.0E0
ENM2 = NA - 1
IF (ENM2 .EQ. 0) GO TO 795

-C  .......... FOR I=EN-2 STEP -1 UNTIL 1 DO -- ..........

DO 790 II = 1, ENM2
I = NA - II
W = BETM * A(I,I) - ALMR * B(I,I)
W1 = -ALMI * B(I,I)
RA = 0.0E0
SA = 0.0E0

DO 760 J = M, EN
X = BETM * A(I,J) - ALMR * B(I,J)
X1 = -ALMI * B(I,J)
RA = RA + X * B(J,NA) - X1 * B(J,EN)
SA = SA + X * B(J,EN) + X1 * B(J,NA)
CONTINUE

C  .......... FOR I=EN-1 STEP -1 UNTIL 1 DO -- ..........

IF (I .EQ. 1 .OR. ISW .EQ. 2) GO TO 770
IF (BETM * A(I,I-1) .EQ. 0.0E0) GO TO 770
ZZ = W
Z1 = W1
R = RA
S = SA
ISW = 2
GO TO 790

770  M = I
IF (ISW .EQ. 2) GO TO 780

C  .......... COMPLEX 1-BY-1 BLOCK ..........

EISP7371  EISP7372  EISP7373  EISP7374  EISP7375  EISP7376  EISP7377  EISP7378  EISP7379  EISP7380  EISP7381  EISP7382  EISP7383  EISP7384  EISP7385  EISP7386  EISP7387  EISP7388  EISP7389  EISP7390  EISP7391  EISP7392  EISP7393  EISP7394  EISP7395  EISP7396  EISP7397  EISP7398  EISP7399  EISP7400  EISP7401  EISP7402  EISP7403  EISP7404  EISP7405  EISP7406  EISP7407  EISP7408  EISP7409  EISP7410  EISP7411  EISP7412  EISP7413  EISP7414  EISP7415  EISP7416  EISP7417  EISP7418  EISP7419  EISP7420  EISP7421  EISP7422  EISP7423  EISP7424  EISP7425  EISP7426  EISP7427  EISP7428  EISP7429  EISP7430
```
TR = -RA
TI = -SA
DR = W
DI = W1

C ............ COMPLEX DIVIDE (T1,T2) = (TR, TI) / (DR, DI) ............

775 IF (ABS(DI) .GT. ABS(DR)) GO TO 777
RR = DI / DR
D = DR + DI * RR
T1 = (TR + TI * RR) / D
T2 = (TI - TR * RR) / D
GO TO (787,782), ISW

CALL GOTOER

777 RR = DR / DI
D = DR * RR + DI
T1 = (TR * RR + TI) / D
T2 = (TI * RR - TR) / D
GO TO (787,782), ISW

CALL GOTOER

C ............ COMPLEX 2-BY-2 BLOCK ............

780 X = BETM * A(I,I+1) - ALMR * B(I,I+1)
X1 = -ALMI * B(I,I+1)
Y = BETM * A(I+1,I)
TR = Y * RA - W * R + W1 * S
TI = Y * SA - W * S - W1 * R
DR = W * ZZ - W1 * Z1 - X * Y
DI = W * Z1 + W1 * ZZ - X1 * Y
IF (DR .EQ. 0.0E0 .AND. DI .EQ. 0.0E0) DR = EPSB
GO TO 775

782 B(I+1,NA) = T1
B(I+1,EN) = T2
ISW = 1
IF (ABS(Y) .GT. ABS(W) + ABS(W1)) GO TO 785

TR = -RA - X * B(I+1,NA) + X1 * B(I+1,EN)
TI = -SA - X * B(I+1,EN) - X1 * B(I+1,NA)
GO TO 773

785 T1 = (-R - ZZ * B(I+1,NA) + Z1 * B(I+1,EN)) / Y
T2 = (-S - ZZ * B(I+1,EN) - Z1 * B(I+1,NA)) / Y
B(I,NA) = T1
B(I,EN) = T2

790 CONTINUE

-C ............ END COMPLEX VECTOR ............

795 ISW = 3 - ISW
800 CONTINUE

-C ............ END BACK SUBSTITUTION.

-C TRANSFORM TO ORIGINAL COORDINATE SYSTEM.

-C FOR J=N STEP -1 UNTIL 1 DO -- ............

DO 880 JJ = 1, N
J = N + 1 - JJ

-C DO 880 I = 1, N
ZZ = 0.0E0

-C DO 860 K = 1, J
ZZ = ZZ + Z(I,K) * B(K,J)

-C Z(I,J) = ZZ
880 CONTINUE

-C ............ NORMALIZE SO THAT MODULUS OF LARGEST
-C COMPONENT OF EACH VECTOR IS 1.
-C (ISW IS 1 INITIALLY FROM BEFORE) ............
```
DO 950 J = 1, N
D = 0.0E0
IF (ISW .EQ. 2) GO TO 920
IF (ALFI(J) .NE. 0.0E0) GO TO 945
DO 890 I = 1, N
  IF (ABS(Z(I,J)) .GT. D) D = ABS(Z(I,J))
890 CONTINUE
DO 900 I = 1, N
  Z(I,J) = Z(I,J) / D
900 CONTINUE
DO 930 I = 1, N
  R = ABS(Z(I,J-1)) + ABS(Z(I,J))
  IF (R .NE. 0.0E0) R = R * SQRT((Z(I,J-1)/R)**2 + (Z(I,J)/R)**2)
930 CONTINUE
DO 940 I = 1, N
  Z(I,J-1) = Z(I,J-1) / D
  Z(I,J) = Z(I,J) / D
940 CONTINUE
ISW = 3 - ISW
CONTINUE
RETURN
END

ROUTINE NAME - PF266=RGG
FROM EISPACK

LATEST REVISION - AUGUST 1, 1984
COMPUTER SCIENCES CORP., HAMPTON, VA.

PURPOSE - THIS SUBROUTINE CALLS THE RECOMMENDED
SEQUENCE OF SUBROUTINES FROM THE EIGENSYSTEM
SUBROUTINE PACKAGE (EISPACK) TO FIND THE
EIGENVALUES AND EIGENVECTORS (IF DESIRED) FOR
THE REAL GENERAL GENERALIZED EIGENPROBLEM
AX = (LAMBDA)BX.

USAGE - CALL RGG(NM,N,A,B,ALFR,ALFI,BETA,MATZ,Z,IERR)

ARGUMENTS
NM - ON INPUT NM MUST BE SET TO THE ROW DIMENSION
OF THE TWO-DIMENSIONAL ARRAY PARAMETERS AS
DECLARED IN THE CALLING PROGRAM DIMENSION
STATEMENT.

N - ON INPUT N IS THE ORDER OF THE MATRICES A
AND B.

A - ON INPUT A CONTAINS A REAL GENERAL MATRIX.
MUST BE OF DIMENSION NM X N.
B - ON INPUT B CONTAINS A REAL GENERAL MATRIX. MUST BE OF DIMENSION NM X N.
ALFR - ON OUTPUT ALFR CONTAINS THE REAL PART OF THE NUMERATORS OF THE EIGENVALUES. MUST BE OF DIMENSION N.
ALFI - ON OUTPUT ALFI CONTAINS THE IMAGINARY PART OF THE NUMERATORS OF THE EIGENVALUES. MUST BE OF DIMENSION N.
BETA - ON OUTPUT BETA CONTAINS THE DENOMINATORS OF THE EIGENVALUES, WHICH ARE THUS GIVEN BY THE RATIOS (ALFR+I*ALFI)/BETA. COMPLEX CONJUGATE PAIRS OF EIGENVALUES APPEAR CONSECUTIVELY WITH THE EIGENVALUE HAVING THE POSITIVE IMAGINARY PART FIRST. MUST BE OF DIMENSION N.
MATZ - ON INPUT MATZ IS AN INTEGER VARIABLE SET EQUAL TO ZERO IF ONLY EIGENVALUES ARE DESIRED. OTHERWISE IT IS SET TO ANY NON-ZERO INTEGER FOR BOTH EIGENVALUES AND EIGENVECTORS.
IERR - ON OUTPUT IERR IS AN INTEGER OUTPUT VARIABLE SET EQUAL TO AN ERROR COMPLETION CODE DESCRIBED IN THE DOCUMENTATION FOR QZIT PF261). THE NORMAL COMPLETION CODE IS ZERO.
REQUIRED ROUTINES - PF260=QZHES,PF261=QZIT,PF262=QZVAL,PF263=QZVECR GG HC318=EPSLON

REMARKS 1. REFERENCES FROM THE EISPACK PACKAGE OF EIGENSYSTEM ROUTINES.

2. SUBROUTINE RGG IS A DRIVER ROUTINE WHICH CALLS ROUTINES QZHES(PF260), QZIT(PF261), QZVAL(PF262), AND QZVEC(PF263).
QZHES(PF260) ACCEPTS A PAIR OF REAL GENERAL MATRICES AND REDUCES ONE OF THEM TO UPPER HESSENBERG FORM AND THE OTHER TO UPPER TRIANGULAR FORM USING ORTHOGONAL TRANSFORMATIONS.
EXAMPLE:

PROGRAM TRGG(OUTPUT, TAPE6=OUTPUT)

DIMENSION A(5,5), B(5,5), ALFR(5), ALFI(5), BETA(5), Z(5,5)

N = 5
NM = 5
MATZ = 1

DATA A /10., 2., 3., 2*1., 2., 12., 1., 2., 1., 3., 1., 11.,
* 1., -1., 1., 2., 1., 9., 3*1., -1., 1., 15. /

DATA B /12., 1., -1., 2., 2*1., 14., 1., -1., 1., -1., 1.,
* 16., -1., 1., 2., -1., -1., 12., -1., 3*1., -1., 11. /

CALL RGG(NM, N, A, B, ALFR, ALFI, BETA, MATZ, Z, IERR)

WRITE (6,99) IERR
WRITE (6,100) ALFR, ALFI, BETA, ((Z(I,J), I=1,5) ,J=1,5)

FORMAT(1H1,7HIERR = ,I4)
FORMAT(1H0,7HALFR = /1H ,5(G8.2,2X)/
* 8H0ALFI = /1H ,5(G8.2,2X)/
* 8H0BETA = /1H ,5(G8.2,2X)/
* 5H0Z = /5(1H ,5(G8.2,2X))/

STOP
END

OUTPUT:

IERR = 0
ALFR = 15. 7.2 16. 10. 8.6
ALFI = 0. 0. 0. 0. 0.
SUBROUTINE diverg(NM, N, A, B, ALFR, ALFI, BETA, MATZ, Z, IERR)

IMPLICIT REAL*8 (A-H,O-Z)
INTEGER NM, N, IERR, MATZ
REAL*8 A(NM, N), B(NM, N), ALFR(N), ALFI(N), BETA(N), Z(NM, N)
LOGICAL TF

ZERO = 0.0E+00
IF (N .LE. NM) GO TO I0
IERR = I0 * N
GO TO 50

10 IF (MATZ .NE. 0) GO TO 20

C  .......... FIND EIGENVALUES ONLY ............
TF = .FALSE.
CALL QZHES(NM, N, A, B, TF, Z)
CALL QZIT(NM, N, A, B, zero , TF, Z, IERR)
CALL QZVAL(NM, N, A, B, ALFR, ALFI, BETA, TF, Z)
GO TO 50

C  .......... FIND BOTH EIGENVALUES AND EIGENVECTORS ............
20 TF = .TRUE.
CALL QZHES(NM, N, A, B, TF, Z)
CALL QZIT(NM, N, A, B, zero , TF, Z, IERR)
CALL QZVAL(NM, N, A, B, ALFR, ALFI, BETA, TF, Z)
IF (IERR .NE. 0) GO TO 50
CALL QZVEC(NM, N, A, B, ALFR, ALFI, BETA, Z)
GO TO 50

RETURN

C** THIS PROGRAM VALID ON FTN4 AND FTN5 **
END

SUBROUTINE gotoer
WRITE (6, I0)
RETURN

I0 FORMAT('there is an error in calculating subroutine')