To the Graduate Council:

I am submitting herewith a thesis written by Gypsy Lynn Tweed entitled "Global Nonlinear Analysis Code for Nuclear Systems". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Nuclear Engineering.

Rafael B. Perez, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Vice Provost
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at The University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Gypsy Lynn Tweed

Date Sept. 27, 1988

GLOBAL NONLINEAR ANALYSIS CODE FOR NUCLEAR SYSTEMS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Gypsy Lynn Tweed

December 1988

In memory of

Mr. Joe B. Waddell, Sr.

ACKNOWLEDGEMENTS

There are many people in the past and present who have supported any endeavor the author chose to pursue and to all of them she shall always be eternally grateful. However, this acknowledgement is dedicated to those who have made a significant contribution, directly or indirectly, to her thesis work.

She wishes thank each member of her family whose continuous encouragement and support has always given her an extra boost when she needed it. The next three gentlemen have supported her not only professionally and academically, but also personally. To Dr. Rafael Perez (her thesis advisor), Dr. Jack Harvey, and Dr. Mark Smith, the author extends her best wishes and a very special, heartfelt thank you. A special thank you to Dr. Dan G. Cacuci for providing the theoretical background for this research and initial support through the Engineering Physics and Mathematics Division (EPMD). Support from Dr. Jim White and the ACTO program of the Instrumentation and Controls Division is also gratefully acknowledged. She also expresses her appreciation to Dr. Jose March–Leuba for his contribution to her thesis. She also wishes to recognize the efforts of the three other members of the her thesis committee, Dr. T. W. Kerlin (NE Dept.), Dr. Ohannes Karakashian (Math Dept.), and Dr. L. F. Miller (NE Dept.). Also, she must thank the faculty of the University of Tennessee Nuclear Engineering Department and the members of the EPM Division at ORNL, for their cooperation and support during her two years of study and assignment in EPMD. Another individual she would like to thank, is Dr. Yousry Azmy of ORNL for his suggestions and assistance with her thesis work. A special note of graditude is extended to Mrs. Dawn Human for her technical advice in the typing of this thesis and to Dr. Jeff O. Johnson for his suggestions on the drafting of this thesis.

Finally, to the one who gave her strength in her weakest moments, and who has always carried her through any crisis. "Through him all things are possible." Thank you, Lord.

ABSTRACT

This work incorporates into a computer code a recently developed formalism for globally computing the critical parameters of nonlinear problems: bifurcations, limit points, and extrema (maxima and minima). The local sensitivities (i.e., first–order derivatives) of the system's state variables (e.g., fluxes, power, temperatures) at any point in the system's phase space are also determined. After initially testing the code with a simple unconstrained nonlinear optimization problem, the code was expanded to handle more sophisticated systems with active or inactive constraints.

There are two test problems used to verify the code's capabilities. One is an unconstrained nonlinear BWR model consisting of a system of five nonlinear equations. This problem was chosen in order to verify the code's ability to locate the first–order bifurcations of this BWR model. The second test problem is a simple constrained optimization problem that is used to test the inactive/active constraint option implemented into the code. The results obtained from these test problems reveal that the code can accomodate active or inactive constrained optimization problems accurately and efficiently.

TABLE OF CONTENTS

CHAPTER PAGE

I. INTRODUCTION 1

II. BRIEF MATHEMATICAL DESCRIPTION OF THE GENERAL METHODOLOGY 5

2.1 Mathematical Descriptions and Definitions 5

2.2 Selection of Nominal Values ($u_\circ$) and Calculation of $F(u_\circ)$ and $F'(u_\circ)$ 8

2.3 Derivation of Newton's Method 10

2.4 Solving $F'(v^\nu)a = F(u_\circ)$ (by using the matrices A or B) . . . 14

2.5 Solving $F'(v^\nu)b = G(v^\nu, \xi^\nu)$ (by using the matrices A or B) . . 15

III. APPLICATIONS 17

3.1 Lumped–Parameter Model of a Boiling Water Reactor (BWR) . 17

3.2 Results for the BWR Test Problem 22

3.3 Constrained Optimization Test Problem 23

3.4 Results for the Constrained Optimization Test Problem . . . 26

IV. CONCLUSIONS AND RECOMMENDATIONS 27

LIST OF REFERENCES 29

APPENDIX 31

Fortran Source Code 33

VITA 75

CHAPTER I

INTRODUCTION

The treatment of nonlinearities which occur in the modelling of most complex physical systems has increased the demand for the development of more powerful methods of computational analysis. Evidence of these nonlinearities can be found in areas such as optimization of nuclear systems and sensitivity analysis. Since nonlinear problems, in general, are difficult to solve, the majority of existing methods involve linearizations. The resulting solutions from these linearized schemes, however, are valid locally, but not globally. Furthermore, as has been shown, for example in References 3 and 5, these linearized methods may completely miss such critical points as bifurcations and limit cycles.

Therefore, a method has been developed by Cacuci[1] based on homotopy theory, (generally refered to as a continuation method) to obtain global results without linearizations. This method is briefly described in Chapter II.

The first significant feature of this method is the recasting of a general mathematical problem composed of three parts: (a) nonlinear constrained optimization, (b) computation of nonlinear eigenvalues and bifurcation points, and (c) sensitivity analysis into a fixed–point problem of the form $F[u(s), \lambda(s)] = 0$.

The second significant feature is the solution of this fixed–point problem by a continuation method (i.e., homotopy method) as developed by Keller[4,5,6]. This homotopy path, designated as $\Gamma(s) = [u(s), \lambda(s)]$, must be constructed such that $G(u, \lambda) \equiv F[u(s)] - \lambda(s)F(u^\circ) = 0$ on the path, (see Figure 1.). The path is thus followed from the initial point $(u^\circ, 1)$ to the final point $[u(s_F), \lambda(s_F)]$, determining along the way the arc parameter values s_i where $\lambda(s_i)$ vanishes, and the values s_j where the Jacobian F'_u vanishes. The resulting values $u(s_i)$ yield the critical

points of F and the extrema points of the response R, while the values $u(s_j)$ yield the limit and bifurcation points.

The code was initially tested by D. G. Cacuci and J. L. Lucius (private communication) with an unconstrained nonlinear optimization problem using the nonlinear equations, $\mathrm{N}(\phi, \alpha)$,

$$N_1 \equiv \phi_1 + \phi_2 - \alpha_1\alpha_2 - \alpha_1\alpha_3 - \alpha_2\alpha_3 - \alpha_1 - \alpha_2 - \alpha_3 - 4 = 0,$$

$$N_2 \equiv \phi_1\phi_2 - (\alpha_1 + \alpha_2 + \alpha_3)(\alpha_1\alpha_2 + \alpha_2\alpha_3 + \alpha_3\alpha_1) + 4 = 0,$$

and a response

$$R = \phi_1^3 - 3\phi_1\phi_2 + 3\alpha_1\alpha_2\alpha_3 - 15\alpha_2 - 288\alpha_3 + 500.$$

where α_1, α_2, and α_3 are system parameters and ϕ_1 and ϕ_2 are the state variables. All of the eight extrema at the points $(\alpha_1, \alpha_2, \alpha_3) = (\pm 2, \pm 3, \pm 10)$ were correctly identified.

As the objective of this thesis, the code was expanded to handle more sophisticated systems with active or inactive constraints. The first test problem used in this thesis was a nonlinear dynamic BWR model. This problem was chosen due to the series of bifurcations that the system is known to generate. Successive bifurcations occur when k, which plays the role of a feedback gain parameter in the system, is varied. To verify this problem the code was extended to reproduce the first bifurcation point reported by the time–step procedure of J. March-Leuba, Cacuci, and Perez.[2] The new method (code) presented in this thesis has the advantage of solving this type of system efficiently, and in considerably less time than the one previously reported.

Although the above BWR problem does not call for constraints, by implementing a constraint option, the code becomes easily applicable to the many optimization problems which require constraints on certain variables such as design specifications. Therefore, this work also uses a simple constrained optimization problem where $g(\alpha) \geq 0$ to check the constraint option. Chapter III contains the

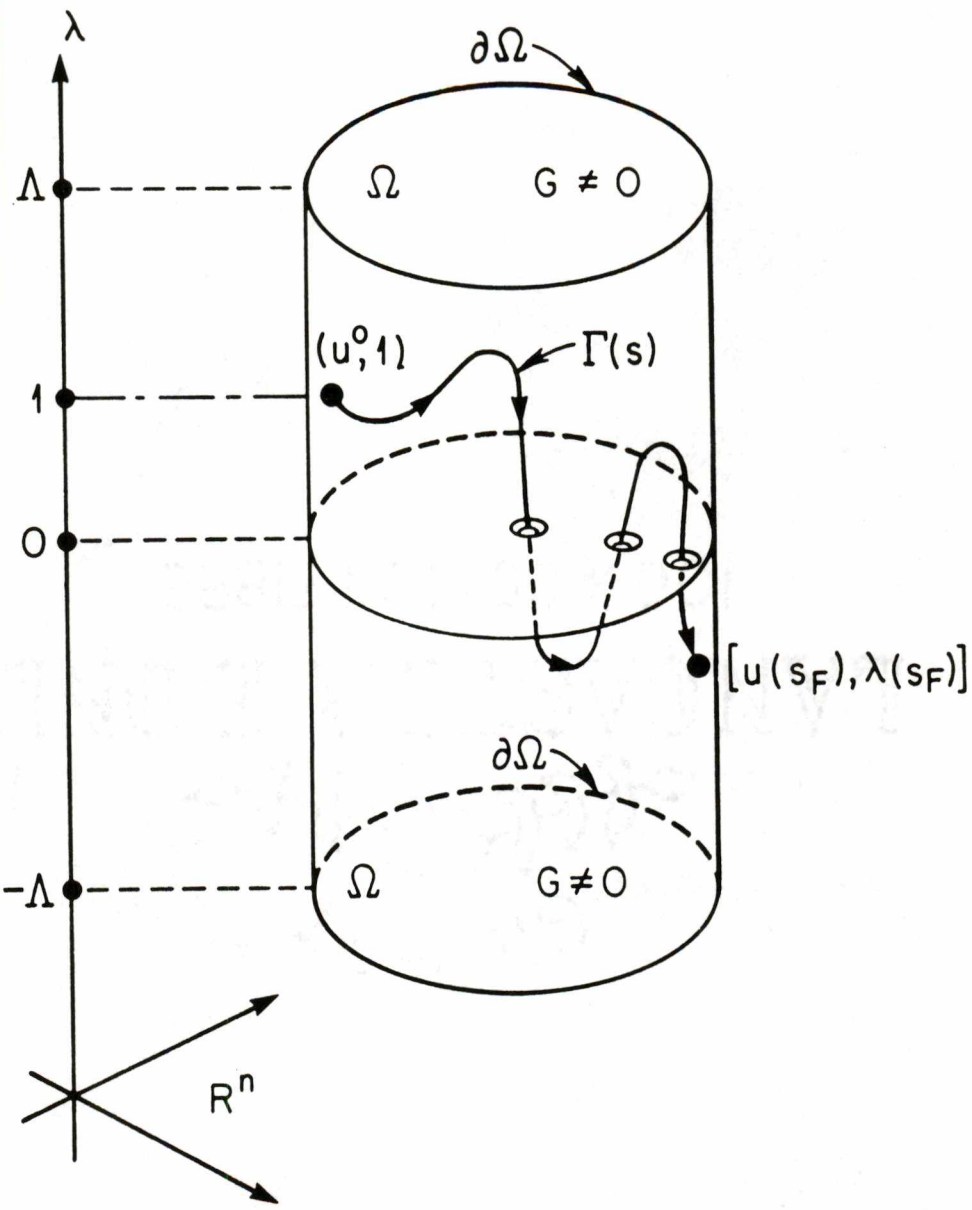


Figure 1. The homotopy path $\Gamma(s) = [u(s), \lambda(s)]$.

descriptions of and results for these two test problems. Chapter IV presents conclusions drawn from this research and recommendations for further improvements to the code, as well as possibilities for extending the scope of the method. Appendix A contains the fortran listing of the code.

In summary, this thesis outlines a computational analysis code developed to compute globally the critical points of nonlinear problems such as bifurcations, limit points, saddle points, extrema (maxima and minima). In addition, this code also calculates the local sensitivities (i.e., first–order derivatives) of the system's state variables (e.g., fluxes, power, temperature) at any point in the system's phase space. The results obtained from the application of the two test problems verified this new method, and revealed that the extensions to the code actually enhanced the method's scope of analysis.

CHAPTER II

BRIEF MATHEMATICAL DESCRIPTION OF THE GENERAL METHODOLOGY

This chapter outlines the general methodology and analysis involved in the set up of the code. This methodology has been developed by D. G. Cacuci[1], whose notation we follow below; the reader is referred to his work for a complete description. This methodology is designed for finding critical points of a response associated with a system, and if the system is nonlinear, find the possible bifurcation and limit points of the system. The method cannot presently be used for finding higher order bifurcation points, or cascades of bifurcations.

2.1 – Mathematical Descriptions and Definitions

First, we introduce the following defintions:

$x = (x_1, \ldots, x_n)^T$; $x \in R^n$, vector of independent variables;

$\phi(x) = (\phi_1(x), \ldots, \phi_m(x))^T$; $\phi : D_\phi \subset R^n \rightarrow R^m$, vector of dependent state variables;

$\alpha = (\alpha_1, \ldots, \alpha_i)^T$; $\alpha \subset R^i$, vector of system parameters;

$N(\phi, \alpha) = (N_1(\phi, \alpha), \ldots, N_m(\phi, \alpha))^T$; $N : D_N \subset D_\phi \times R^i \rightarrow R^m$, nonlinear equations (including boundary and initial conditions);

$g(\alpha) = (g_1(\alpha), \ldots, g_k(\alpha))^T$; $D_g \subset R^i \rightarrow R^k$, inequality/equality constraints;

$y = (y_1, \ldots, y_m)^T$, adjoint functions corresponding to inactive constraints $(g(\alpha) < 0)$;

$z = (z_1, \ldots, z_k)^T$, adjoint-like functions corresponding to active constraints $(g(\alpha) \geq 0)$.

Define the system's response as

$$R(\phi(x), \alpha); \quad R : D_R \subset D_\phi \times R^i \to R^1,$$

with

$$\nabla_1 R = \left(\frac{\partial R}{\partial \phi_1}, \ldots, \frac{\partial R}{\partial \phi_m} \right)^T, \text{ gradient in the } \phi \text{ direction;}$$

$$\nabla_2 R = \left(\frac{\partial R}{\partial \alpha_1}, \ldots, \frac{\partial R}{\partial \alpha_i} \right)^T, \text{ gradient in the } \alpha \text{ direction.}$$

Next, define the matrix quantities (Jacobian matrices),

$$\nabla_1 N = \left[N'_\phi(\phi, \alpha) \right] = \begin{bmatrix} \frac{\partial N_1}{\partial \phi_1} & \cdots & \frac{\partial N_m}{\partial \phi_1} \\ \vdots & & \vdots \\ \frac{\partial N_1}{\partial \phi_m} & \cdots & \frac{\partial N_m}{\partial \phi_m} \end{bmatrix}, \tag{2-1}$$

$$\nabla_2 N = \left[N'_\alpha(\phi, \alpha) \right] = \begin{bmatrix} \frac{\partial N_1}{\partial \alpha_1} & \cdots & \frac{\partial N_m}{\partial \alpha_1} \\ \vdots & & \vdots \\ \frac{\partial N_1}{\partial \alpha_i} & \cdots & \frac{\partial N_m}{\partial \alpha_i} \end{bmatrix}, \tag{2-2}$$

$$\nabla g = \left[g'(\alpha) \right] = \begin{bmatrix} \frac{\partial g_1}{\partial \alpha_1} & \cdots & \frac{\partial g_k}{\partial \alpha_1} \\ \vdots & & \vdots \\ \frac{\partial g_1}{\partial \alpha_i} & \cdots & \frac{\partial g_k}{\partial \alpha_i} \end{bmatrix}. \tag{2-3}$$

Then define the Lagrangian $L(\phi, \alpha)$ as

$$\begin{aligned} L(\phi, \alpha) &= R(\phi, \alpha) + y^T N(\phi, \alpha) + z^T g(\alpha), \\ &= R(\phi, \alpha) + \sum_{j=1}^{m} y_j N_j(\phi, \alpha) + \sum_{j=1}^{k} z_j g_j, \end{aligned}$$

and the gradients,

$$L_1 = \nabla_1 R + (\nabla_1 N)y, \tag{2-4}$$

$$L_2 = \nabla_2 R + (\nabla_2 N)y + (\nabla g)z. \tag{2-5}$$

Also define the constraint function,

$$K(g(\alpha), z) = (g + z)^2 \pm g|g| \mp z|z|. \tag{2-6}$$

where the upper or lower sign is taken according to whether the problem is one of maximization or minimization, respectively.

Let $u = (\phi, y, \alpha, z)$, so the stationary points of R subject to

$$\begin{Bmatrix} N(\phi, \alpha) = 0 \\ g(\alpha) \leq 0 \end{Bmatrix},$$

are the solutions u^* of $F(u) = 0$, where

$$F(u) = \begin{pmatrix} L_1 \\ N(\phi, \alpha) \\ L_2 \\ K(g(\alpha), z) \end{pmatrix}. \tag{2-7}$$

This arrangement is to produce symmetries in $F'(u)$ later.

Now we must solve $F(u) = 0$ using "pseudo–arclength continuation" method as presented by Keller.[4,5,6] Equivalent constraints are chosen to ensure that F is globally differentiable. (F is globally differentiable if R and N are twice globally differentiable with respect to ϕ and α.)

The Jacobian $F'(u)$ is needed to use the "pseudo–arclength continuation" algorithm. This Jacobian is obtained as

$$F'(u) = \begin{pmatrix} L_{11} & \nabla_1 N & L_{12} & 0 & 0 \\ (\nabla_1 N)^T & 0 & (\nabla_2 N)^T & 0 & 0 \\ L_{12}^T & \nabla_2 N & L_{22} & \nabla g_I & \nabla g_J \\ 0 & 0 & 2Z_I(\nabla g_I)^T & 0 & 0 \\ 0 & 0 & 0 & 0 & 2C_J \end{pmatrix}, \tag{2-8}$$

where

$$L_{11} = \begin{pmatrix} \frac{\partial^2 R}{\partial \phi_1 \partial \phi_1} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \phi_1 \partial \phi_1} & \cdots & \frac{\partial^2 R}{\partial \phi_m \partial \phi_1} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \phi_m \partial \phi_1} \\ \vdots & & \vdots \\ \frac{\partial^2 R}{\partial \phi_1 \partial \phi_m} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \phi_1 \partial \phi_m} & \cdots & \frac{\partial^2 R}{\partial \phi_m \partial \phi_m} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \phi_m \partial \phi_m} \end{pmatrix}, \quad (2\text{-}9)$$

$$L_{12} = \begin{pmatrix} \frac{\partial^2 R}{\partial \alpha_1 \partial \phi_1} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \alpha_1 \partial \phi_1} & \cdots & \frac{\partial^2 R}{\partial \alpha_i \partial \phi_1} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \alpha_i \partial \phi_1} \\ \vdots & & \vdots \\ \frac{\partial^2 R}{\partial \alpha_1 \partial \phi_m} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \alpha_1 \partial \phi_m} & \cdots & \frac{\partial^2 R}{\partial \alpha_i \partial \phi_m} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \alpha_i \partial \phi_m} \end{pmatrix}, \quad (2\text{-}10)$$

$$L_{22} = \begin{pmatrix} \frac{\partial^2 R}{\partial \alpha_1 \partial \alpha_1} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \alpha_1 \partial \alpha_1} + \sum_{r=1}^{k} z_r \frac{\partial^2 g_r}{\partial \alpha_1 \partial \alpha_1} & \cdots \\ \vdots & \\ \frac{\partial^2 R}{\partial \alpha_1 \partial \alpha_i} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \alpha_1 \partial \alpha_i} + \sum_{r=1}^{k} z_r \frac{\partial^2 g_r}{\partial \alpha_1 \partial \alpha_i} & \cdots \end{pmatrix}$$

$$\begin{pmatrix} \cdots & \frac{\partial^2 R}{\partial \alpha_i \partial \alpha_1} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \alpha_i \partial \alpha_1} + \sum_{r=1}^{k} z_r \frac{\partial^2 g_r}{\partial \alpha_i \partial \alpha_1} \\ & \vdots \\ \cdots & \frac{\partial^2 R}{\partial \alpha_i \partial \alpha_i} + \sum_{j=1}^{m} y_j \frac{\partial^2 N_j}{\partial \alpha_i \partial \alpha_i} + \sum_{r=1}^{k} z_r \frac{\partial^2 g_r}{\partial \alpha_i \partial \alpha_i} \end{pmatrix}. \quad (2\text{-}11)$$

Define the matrix C_J of inactive constraints ($g_j(\alpha) < 0$ and $z_j = 0$) as the diagonal matrix with elements g_j, $j \in$J;

$$C_J = \begin{pmatrix} \bullet & & 0 \\ & g_j & \\ 0 & & \bullet \end{pmatrix}, \quad (2\text{-}12)$$

where J is the set of inactive constraints.

When constraints are active ($g_j(\alpha) \geq 0$ and $z_j \neq 0$) then define the matrix Z_I to be a diagonal matrix with elements z_j, $j \in$I;

$$Z_I = \begin{pmatrix} \bullet & & 0 \\ & z_j & \\ 0 & & \bullet \end{pmatrix}, \tag{2-13}$$

where I is the set of active constraints.

2.2 – Selection of Nominal Values ($u_\circ$) and Calculation of $F(u_\circ)$ and $F'(u_\circ)$

Using $u_\circ = (\phi_\circ, y_\circ, \alpha_\circ, z_\circ)$ as a starting point, the $\alpha_\circ$'s are the only nominal values given. The nominal values ϕ_j° are determined by solving the system $N(\phi_\circ, \alpha_\circ) = 0$. It is always possible to arrange the constraints $g_j(\alpha)$ so that they are all inactive at $\alpha_\circ$, i.e. $g_j(\alpha_\circ) < 0$. Therefore, the nominal values of $z_j^\circ = 0$. And the nominal values for the Lagrange multipliers y_j (adjoint functions), are obtained by solving the adjoint equation,

$$L_1 = \nabla_1 R(\phi_\circ, \alpha_\circ) + [\nabla_1 N(\phi_\circ, \alpha_\circ)]y^\circ = 0. \tag{2-14}$$

However, if the constraints are active, i.e. $g_j(\alpha_\circ) \geq 0$, then $y_j^\circ = 0$ and the z_j°'s can be obtained by solving

$$L_2 = \nabla_2 R(\phi_\circ, \alpha_\circ) + [\nabla g(\alpha_\circ)]z = 0. \tag{2-15}$$

Thus by choosing inactive constraints, $u_\circ = (\phi_\circ, y_\circ, \alpha_\circ, 0)$ is determined and

$$F(u_\circ) = \begin{pmatrix} 0 \\ 0 \\ L_2(u_\circ) \\ 0 \\ 0 \end{pmatrix},$$

where

$$L_2 = \nabla_2 R(\phi_\circ, \alpha_\circ) + [\nabla_2 N(\phi_\circ, \alpha_\circ)]y^\circ.$$

Note: $L_2(u_\circ)$ is the vector of first–order sensitivities. (See Reference 1.)

With the nominal values defined above, we can write $F'(u_\circ)$ as

$$F'(u_\circ) = \begin{pmatrix} L_{11}(u_\circ) & \nabla_1 N(u_\circ) & L_{12}(u_\circ) & 0 \\ [\nabla_1 N(u_\circ)]^T & 0 & [\nabla_2 N(u_\circ)]^T & 0 \\ L_{12}^T(u_\circ) & \nabla_2 N(u_\circ) & L_{22}(u_\circ) & \nabla g_J(\alpha_\circ) \\ 0 & 0 & 0 & 2C_J(\alpha_\circ) \end{pmatrix},$$

and we note that $F'(u_\circ)$ must be nonsingular.

Now to start the pseudo–arclength continuation procedure, we need a point, $[u_\circ, \lambda_\circ] = [u^\circ, 1]$, on the homotopy path $\Gamma(s)$, and the corresponding tangent vector $[\dot{u}_\circ, \dot{\lambda}_\circ]$. The homotopy path is $u(s)$, such that

$$G(u, \lambda) = F[u(s)] - \lambda(s)F(u_\circ) = 0, \tag{2-16}$$

and consequently,

$$F'(u)\dot{u} - \dot{\lambda}F(u_\circ) = 0. \tag{2-17}$$

By making s the arclength along the path $\Gamma(s) = [u(s), \lambda(s)]$, we have the condition

$$\|\dot{u}(s)\|^2 + \dot{\lambda}^2(s) = 1. \tag{2-18}$$

Since $[u_\circ, \lambda_\circ]$ must satisfy equations (2-16), (2-17), and (2-18) at $s = 0$, we obtain

$$\dot{u}(0) = \dot{\lambda}(0)\,[F'(u_\circ)]^{-1}F(u_\circ),$$

$$\dot{\lambda}(0) = \pm\,\{1 + \|[F'(u_\circ)]^{-1}F(u_\circ)\|^2\}^{1/2},$$

where the sign is such that $\dot{u}(0)$ points into Ω from $u_\circ \in \partial\Omega$. Here, $u \in \Omega$, and $u_\circ \in \partial\Omega$, and $F(u)$ must satisfy on $\partial\Omega$ the "Smale boundary conditions";

(a) $F'(u)$ is nonsingular on $\partial\Omega$,

and either

(b) $[F'(u)]^{-1}\,F(u)$ points into Ω for all $u \in \partial\Omega$,

or

(c) $[F'(u)]^{-1}\,F(u)$ points out of Ω for all $u \in \partial\Omega$.

To seek another point $[v, \xi]$ on Γ use Newton's Method to solve

$$G(v, \xi) = F(v) - \xi F(u_\circ) = 0,$$

$$M(v, \xi, \sigma) = \dot{u}_\circ^T (v - u_\circ) + \dot{\lambda}_\circ (\xi - \lambda_\circ) - (\sigma - s_\circ) = 0.$$

Note: Although in principle $s_\circ$ can be arbitrary, we set $s_\circ = 0$ in this work.

2.3 – Derivation of Newton's Method

The following is a derivation of Newton's method as it pertains to the above system of equations. Beginning with the general form

$$x^{k+1} = x^k - [\mathcal{F}'(x^k)]^{-1} \mathcal{F}(x^k),$$

therefore,

$$\mathcal{F}'(x^k)(x^{k+1} - x^k) = -\mathcal{F}(x^k).$$

So, let

$$\mathcal{F} = \begin{pmatrix} \overbrace{F(v) - \xi F(u_\circ)}^{G(v,\xi)} \\ \underbrace{\dot{u}_\circ^T (v - u_\circ) + \dot{\lambda}_\circ (\xi - \lambda_\circ) - \sigma}_{M(v,\xi,\sigma)} \end{pmatrix},$$

and therefore obtain,

$$\mathcal{F}' = \begin{pmatrix} F'(v) & -F(u_\circ) \\ \dot{u}_\circ^T & \dot{\lambda}_\circ \end{pmatrix}.$$

Now by letting $x = \begin{pmatrix} v \\ \xi \end{pmatrix}$ and $\nu = k$ we get,

$$x^{\nu+1} - x^\nu = \begin{pmatrix} v^{\nu+1} - v^\nu \\ \xi^{\nu+1} - \xi^\nu \end{pmatrix} = \begin{pmatrix} \partial v^\nu \\ \partial \xi^\nu \end{pmatrix}.$$

Consequently, the final form of Newton's method to be applied in this work is as follows,

$$\begin{pmatrix} F'(v^\nu) & -F(u_\circ) \\ \dot{u}_\circ^T & \dot{\lambda}_\circ \end{pmatrix} \begin{pmatrix} \partial v^\nu \\ \partial \xi^\nu \end{pmatrix} = -\begin{pmatrix} \overbrace{F(v^\nu) - \xi^\nu F(u_\circ)}^{G(v^\nu,\xi^\nu)} \\ \underbrace{\dot{u}_\circ^T (v^\nu - u_\circ) + \dot{\lambda}_\circ (\xi^\nu - \lambda_\circ) - \sigma}_{M(v^\nu,\xi^\nu,\sigma)} \end{pmatrix}.$$

It is impractical to solve this system using $F'(v^\nu)$ as it stands; the dimensions can become very large. Things may become more complicated by the fact that $F'(v^\nu)$ can become singular, although Newton's method can still be shown to converge quadratically at such points. Therefore, we chose a different approach. Solve this system for ∂v^ν and $\partial \xi^\nu$,

$$F'(v^\nu)\partial v^\nu - F(u_\circ)\partial \xi^\nu = -G(v^\nu, \xi^\nu),$$

$$\dot{u}_\circ^T \partial v^\nu + \dot{\lambda}_\circ \partial \xi^\nu = -M(v^\nu, \xi^\nu, \sigma),$$

$$\partial v^\nu = -[F'(v^\nu)]^{-1} G(v^\nu, \xi^\nu) + [F'(v^\nu)]^{-1} F(u_\circ)\partial \xi^\nu, \tag{2-19}$$

$$\dot{u}_\circ^T \partial v^\nu + \dot{\lambda}_\circ \partial \xi^\nu = -M(v^\nu, \xi^\nu, \sigma). \tag{2-20}$$

Then by defining

$$a = [F'(v^\nu)]^{-1} F(u_\circ),$$

$$b = [F'(v^\nu)]^{-1} G(v^\nu, \xi^\nu),$$

and rearranging equations (2-19) and (2-20) we get,

$$\partial v^\nu = \partial \xi^\nu a - b, \tag{2-21}$$

$$\partial \xi^\nu = [\dot{u}_\circ^T b - M(v^\nu, \xi^\nu, \sigma)]/[\dot{u}_\circ^T a + \dot{\lambda}_\circ]. \tag{2-22}$$

The initial iterate is

$$\begin{Bmatrix} v_\circ = u_\circ + \sigma \dot{u}_\circ \\ \xi_\circ = \lambda_\circ + \sigma \dot{\lambda}_\circ \end{Bmatrix},$$

where σ is estimated from

$$|\sigma| = \{\| \frac{1}{\Delta^2} [F(u_\circ + \Delta \dot{u}_\circ) - 2F(u_\circ) + F(u_\circ - \Delta \dot{u}_\circ)] \| \, \mathcal{K}(\mathcal{A})\}^{-1/2}, \tag{2-23}$$

where Δ is a small increment (.01), and $\mathcal{K}(\mathcal{A})$ is the condition number of $\mathcal{A}$, i.e.,

$$\mathcal{K}(\mathcal{A}) = \|\mathcal{A}^{-1}\| \, \|\mathcal{A}\|,$$

and

$$\mathcal{A} \equiv \begin{bmatrix} F'(u_\circ) & -F(u_\circ) \\ \dot{u}_\circ^T & \dot{\lambda}_\circ \end{bmatrix}$$

is nonsingular. After obtaining ∂v^ν from (2-18) and $\partial \xi^\nu$ from (2-19), calculate the next iterates,

$$v^{\nu+1} = v^\nu + \partial v^\nu,$$

$$\xi^{\nu+1} = \xi^\nu + \partial \xi^\nu.$$

Continue until convergence is achieved, i.e., until ∂v^ν and $\partial \xi^\nu$ are smaller than some prespecified criterion ϵ (10^{-8}). After convergence, we have obtained $[v(\sigma), \xi(\sigma)]$.

Then use the last iterate (a) of the equation

$$\text{or} \quad \begin{cases} A(v)a = \begin{pmatrix} 0 \\ 0 \\ L_2(u_\circ) \end{pmatrix} & \text{– if all constraints are inactive} \\ B(v)a = \begin{pmatrix} 0 \\ 0 \\ L_2(u_\circ) \\ 0 \end{pmatrix} & \text{– otherwise} \end{cases}$$

to form

$$\dot{\xi} = \pm\, [1 + \|a\|^2]^{-1/2},$$

$$\dot{v} = \dot{\xi}\, a,$$

where the sign of $\dot{\xi}$ is chosen such that

$$\dot{u}_\circ^T \dot{v} + \dot{\lambda}\, \dot{\xi} > 0.$$

Note: $A(v)$ and $B(v)$ are submatrices of $F'(v)$.

Now replace $[u_\circ, \lambda_\circ]$ by $[v(\sigma), \xi(\sigma)]$ and $[\dot{u}_\circ, \dot{\lambda}_\circ]$ by $[\dot{v}(\sigma), \dot{\xi}(\sigma)]$ and repeat the iterative procedure (ν) to obtain a new point $[v, \xi]$ on $\Gamma(s)$. When λ changes sign between two points $[v(\sigma_j), \xi(\sigma_j)]$ and $[v(\sigma_{j+1}), \xi(\sigma_{j+1})]$, then $F(u) = 0$ has a root $u^* = v(\sigma^*)$ such that $\sigma^* \in (\sigma_j, \sigma_{j+1})$ and $\xi(\sigma^*) = 0$. Similarly, when $\dot{\lambda}$ changes sign between $[\dot{v}(\sigma_j), \dot{\xi}(\sigma_j)]$ and $[\dot{v}(\sigma_{j+1}), \dot{\xi}(\sigma_{j+1})]$, then $det\ F'(u) = 0$ (i.e. a limit

point exists) and $\xi_i(\sigma_j)$ should be replaced by $\xi_i'(\sigma_j) = [\dot{\lambda} + u_o^T a]^{-1}$ when using the secant method to determine σ^*.

Now calculate $\sigma^* \in (\sigma_j, \sigma_{j+1})$ using the secant method and $\xi(\sigma_j)\,\xi(\sigma_{j+1}) < 0$ as,

$$\sigma_{k+1} = \sigma_k - \xi_k \frac{\sigma_k - \sigma_{k-1}}{\xi_k - \xi_{k-1}},$$

where $\xi_k = \xi(\sigma_k)$. The sequence $\{\sigma_k\}$, converges to σ^* such that $v(\sigma^*)$ is a root, and $\xi(\sigma^*) = 0$ or $\dot{v}(\sigma^*)$ is a limit point and $\dot{\xi}(\sigma^*) = 0$.

The secant method above can be modified to insure convergence by coupling it to *regular falsi*. In this scheme, retain the current iterate σ_j and the most current iterate σ_k such that $\xi_j \xi_k < 0$. Then, if the next iterate $\sigma_{k+1} \notin [\sigma_j, \sigma_k]$, discard this iterate σ_{k+1}, and use instead a σ_{k+1} obtained from

$$\sigma_{k+1} = \frac{\xi_j \sigma_k - \xi_k \sigma_j}{\xi_j - \xi_k}.$$

Once σ^* is obtained (with $\epsilon_1 = 10^{-12}$ as convergence criterion) use as new initial iterates (in the ν iteration)

$$v_o(\sigma_{j+1}) = \theta\, v(\sigma_j) + (1-\theta)\, v(\sigma_k),$$

$$\xi_o(\sigma_{j+1}) = 0,$$

where

$$\theta = \frac{\sigma_{j+1} - \sigma_k}{\sigma_j - \sigma_k}.$$

Here k is $j-1$ if σ_{j+1} is determined from the secant method, otherwise, it is as in *regular falsi* above.

2.4 – Solving $F'(v^\nu)a = F(u_o)$ (by using the matrices A or B)

Define the vector a as

$$a = \begin{pmatrix} a_\phi \\ a_y \\ a_\alpha \\ a_I \\ a_J \end{pmatrix}.$$

1) When all constraints are inactive, then solve

$$[A(v^\nu)] \begin{pmatrix} a_\phi \\ a_y \\ a_\alpha \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ L_2(u_\circ) \end{pmatrix},$$

to determine a_ϕ, a_y, and a_α. Here $a_I = 0$ and $a_J = 0$ because $\mathrm{I} = \emptyset$ and $F_J = 0$ (z_j's $= 0$).

2) When some constaints are active ($\mathrm{I} \neq \emptyset$), then solve

$$[B(v^\nu)] \begin{pmatrix} a_\phi \\ a_y \\ a_\alpha \\ a_I \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ L_2(\phi_\circ, y_\circ, \alpha_\circ, z) \\ 0 \end{pmatrix},$$

Repeat for $\nu = 0, 1, \ldots$. Note that the last iterate gives the solution of $F'(v)a = F(u_\circ)$.

2.5 – Solving $F'(v^\nu)b = G(v^\nu, \xi^\nu)$ (by using the matrices A or B)

Define the vector b as

$$b = \begin{pmatrix} b_\phi \\ b_y \\ b_\alpha \\ b_I \\ b_J \end{pmatrix}.$$

Recall that $G(v^\nu, \xi^\nu) = F(v) - \xi F(u_\circ)$; here we define the G vector as follows,

$$G_\phi = L_1(\phi^\nu, y^\nu, \alpha^\nu),$$

$$G_y = N(\phi^\nu, \alpha^\nu),$$

$$G_\alpha = L_2(\phi^\nu, y^\nu, \alpha^\nu, z^\nu) - \xi^\nu L_2(\phi_\circ, y_\circ, \alpha_\circ, z_\circ),$$

$$G_I = K_I(\alpha^\nu, z^\nu),$$

$$G_J = K_J(\alpha^\nu, z^\nu).$$

1) When all constraints are inactive, then solve

$$[A(v^\nu)] \begin{pmatrix} b_\phi \\ b_y \\ b_\alpha \end{pmatrix} = \begin{pmatrix} G_\phi \\ G_y \\ G_\alpha - \frac{1}{2}[\nabla g_m(\alpha^\nu, z^\nu)]^{-1}[C_m(\alpha^\nu, z^\nu)]^{-1} K_m(\alpha^\nu, z^\nu) \end{pmatrix},$$

to obtain b_ϕ, b_y, and b_α. Since I $= \emptyset$ and $J = m$; note that $G_I = \emptyset, G_J = K_m$; so that $b_I \neq \emptyset$ and $b_J = b_m$, where

$$b_m = \frac{1}{2}[C_m(\alpha^\nu, z^\nu)]^{-1} K_m(\alpha^\nu, z^\nu),$$

2) When I constraints are active:

$$[B(v^\nu)] \begin{pmatrix} b_\phi \\ b_y \\ b_\alpha \\ b_I \end{pmatrix} = \begin{pmatrix} G_\phi \\ G_y \\ G_\alpha - \frac{1}{2}[\nabla g_J(\alpha^\nu, z^\nu)] b_J \\ \frac{1}{2}[Z_I(z^\nu)]^{-1} K_I(\alpha^\nu, z^\nu) \end{pmatrix},$$

where

$$b_J = \frac{1}{2}[C_J(\alpha^\nu, z^\nu)]^{-1} K_J(\alpha^\nu, z^\nu).$$

Again, repeat for $\nu = 0, 1, \ldots$; the last iterate gives the solution of $F'(v^\nu)b = G(v^\nu, \xi^\nu)$.

This concludes the basic methodology and mathematical descriptions used in this work. Further details can be found in the works by Cacuci[1] and Keller[4,5,6].

CHAPTER III

APPLICATIONS

This chapter describes the application of this method to two test problems. The first one is a nonlinear dynamic BWR model with inactive constraints, $g(\alpha) < 0$. The objective was to check the code's ability to identify limit points (bifurcations) by reproducing the first bifurcation point previously reported by March–Leuba, Cacuci, and Perez.[2] The second problem is an optimization problem with active constraints, $g(\alpha) \geq 0$. The purpose of this problem was to test the constraint option added to the code.

3.1 – Lumped–Parameter Model of a Boiling Water Reactor (BWR)

To describe the dynamic behavior of this BWR system, the following spatially averaged quantities are selected as dependent variables:

1. excess neutron population $n(t)$
2. excess population of delayed neutron precursors $c(t)$
3. excess fuel temperature $T(t)$
4. relative excess coolant density $\gamma(t)$
5. excess reactivity $\rho(t)$.

Here "excess" signifies departure from the initial steady–state values at $t = 0$ when the reactor is critical.

For $t > 0$, the dependent variables satisfy the following equations:

$$\frac{dn(t)}{dt} = n(t)[\rho(t) - \beta]/\Lambda + \lambda c(t) + \rho(t)/\Lambda, \tag{3-1}$$

$$\frac{dc(t)}{dt} = n(t)\beta/\Lambda - \lambda c(t), \tag{3-2}$$

$$\frac{dT(t)}{dt} = Q[n(t) + H(t)\Delta] - a_3 T(t), \tag{3-3}$$

$$\frac{d^2\gamma(t)}{dt^2} + a_2\frac{d\gamma(t)}{dt} + a_1\gamma(t) = kT(t), \tag{3-4}$$

$$\rho(t) = \Gamma_1\gamma(t) + \Gamma_2 T(t), \tag{3-5}$$

where $H(t)$ is the Heaviside step–function and the various constants are given in Reference 3.

By letting,

$$f(t) = \frac{d\gamma(t)}{dt}$$

equations (3-1) through (3-5) can be rewritten as the following system of five coupled first–order nonlinear ODE's:

$$\frac{dn}{dt} = n(t)[\Gamma_1\gamma(t) + \Gamma_2 T(t) - \beta]/\Lambda + \lambda c(t) + [\Gamma_1\gamma(t) + \Gamma_2 T(t)]/\Lambda, \tag{3-6}$$

$$\frac{dc}{dt} = n(t)\beta/\Lambda - \lambda c(t), \tag{3-7}$$

$$\frac{dT}{dt} = Q[n(t) + H(t)\Delta] - a_3 T(t), \tag{3-8}$$

$$\frac{d\gamma}{dt} = f(t), \tag{3-9}$$

$$\frac{df}{dt} = -a_2 f(t) - a_1\gamma(t) + kT(t). \tag{3-10}$$

Defining the (column) vectors,

$$\phi = (\phi_1, \ldots, \phi_5),$$

$$\alpha = (\alpha_1, \ldots, \alpha_{11}),$$

and

$$N(\phi, \alpha) = (N_1(\phi, \alpha), \ldots, N_5(\phi, \alpha)),$$

where

$$\alpha_1 = \Gamma_1,$$

$$\alpha_2 = \Gamma_2,$$

$$\alpha_3 = \beta,$$

$$\phi_1 = n(t), \qquad \alpha_4 = \Lambda,$$

$$\phi_2 = c(t), \qquad \alpha_5 = \lambda,$$

$$\phi_3 = T(t), \qquad \alpha_6 = Q,$$

$$\phi_4 = \gamma(t), \qquad \alpha_7 = \Delta,$$

$$\phi_5 = f(t), \qquad \alpha_8 = a_3,$$

$$\alpha_9 = a_2,$$

$$\alpha_{10} = a_1,$$

$$\alpha_{11} = k,$$

it follows that equations (3-6) through (3-10) can be written in vector form as

$$N(\phi, \alpha) = 0,$$

where

$$N_1 = \frac{d\phi_1}{dt} - \phi_1[\alpha_1\phi_4 + \alpha_2\phi_3 - \alpha_3]/\alpha_4 - \alpha_5\phi_2 - [\alpha_1\phi_4 + \alpha_2\phi_3]/\alpha_4, \tag{3-11}$$

$$N_2 = \frac{d\phi_2}{dt} - \phi_1\alpha_3/\alpha_4 + \alpha_5\phi_2, \tag{3-12}$$

$$N_3 = \frac{d\phi_3}{dt} - \alpha_6[\phi_1 + \alpha_7 H(t)] + \alpha_8\phi_3, \tag{3-13}$$

$$N_4 = \frac{d\phi_4}{dt} - \phi_5, \tag{3-14}$$

$$N_5 = \frac{d\phi_5}{dt} + \alpha_9\phi_5 + \alpha_{10}\phi_4 - \alpha_{11}\phi_3. \tag{3-15}$$

Consider that the system's response is a functional of the form,

$$R(\phi, \alpha) = \int_0^{t_f} \left(\sum_{j=1}^{5} \phi_j \alpha_{11+j} \right) dt;$$

for this work,

$$R(\phi, \alpha) = \phi_3.$$

For global sensitivity analysis, find the extrema of R and the bifurcation points of $N(\phi, \alpha) = 0$ as the parameter α_{11} varies over ranges specified by the inequality constraints

$$g(\alpha) \leq 0.$$

Next, we determine the expressions of the various components of $F(u)$ (2-7) and $F'(u)$ (2-8) (see pp. 1–4 of Chapter 2). These components are as follows:

1)

$$\nabla_1 N = \frac{\partial N_i}{\partial \phi_j}, \quad i, j = 1, \ldots, 5$$

where $\dfrac{\partial N_i}{\partial \phi_j}$ is the partial derivative of N_i with respect to ϕ_j. Therefore, the matrix

$$\nabla_1 N = \begin{bmatrix} \dfrac{-\alpha_1\phi_4 - \alpha_2\phi_3 + \alpha_3}{\alpha_4} & \dfrac{-\alpha_3}{\alpha_4} & -\alpha_6 & 0 & 0 \\ -\alpha_5 & \alpha_5 & 0 & 0 & 0 \\ \dfrac{-(\phi_1 + 1)\alpha_2}{\alpha_4} & 0 & \alpha_8 & 0 & -\alpha_{11} \\ \dfrac{-(\phi_1 + 1)\alpha_1}{\alpha_4} & 0 & 0 & 0 & \alpha_{10} \\ 0 & 0 & 0 & -1 & \alpha_9 \end{bmatrix}.$$

2)

$$\nabla_2 N = \frac{\partial N_i}{\partial \alpha_j}, \quad i = 1, \ldots, 5, \quad j = 1, \ldots, 11$$

where $\dfrac{\partial N_i}{\partial \alpha_j}$ is the partial derivative of N_i with respect to α_j. Therefore, the matrix

$$\nabla_2 N = \begin{bmatrix} \frac{-\phi_1\phi_4}{\alpha_4} - \frac{\phi_4}{\alpha_4} & 0 & 0 & 0 & 0 \\ \frac{-\phi_1\phi_3}{\alpha_4} - \frac{\phi_3}{\alpha_4} & 0 & 0 & 0 & 0 \\ \frac{\phi_1}{\alpha_4} & \frac{-\phi_1}{\alpha_4} & 0 & 0 & 0 \\ \frac{\phi_1[\alpha_1\phi_4 + \alpha_2\phi_3 - \alpha_3] + \alpha_1\phi_4 + \alpha_2\phi_3}{\alpha_4^2} & \frac{\phi_1\alpha_3}{\alpha_4^2} & 0 & 0 & 0 \\ -\phi_2 & \phi_2 & 0 & 0 & 0 \\ 0 & 0 & -\phi_1 - \alpha_7 & 0 & 0 \\ 0 & 0 & -\alpha_6 & 0 & 0 \\ 0 & 0 & \phi_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & \phi_5 \\ 0 & 0 & 0 & 0 & \phi_4 \\ 0 & 0 & 0 & 0 & -\phi_3 \end{bmatrix}.$$

However, since we consider all the α's except α_{11} to be constant, $\nabla_2 N$ reduces to a row vector, $(0, 0, 0, 0, -\phi_3)$.

3) From equations (2-4) and (2-5), the vectors L_1 and L_2 are defined as

$$L_1 = \begin{pmatrix} \frac{-\alpha_1\phi_4 - \alpha_2\phi_3 + \alpha_3}{\alpha_4} y_1 - \frac{\alpha_3}{\alpha_4} - \alpha_6 y_3 \\ -\alpha_5 y_1 + \alpha_5 y_2 \\ 1.0 - (\phi_1 + 1)\frac{\alpha_2}{\alpha_4} y_1 + \alpha_8 y_3 - \alpha_{11} y_5 \\ -(\phi_1 + 1)\frac{\alpha_1}{\alpha_4} y_1 + \alpha_{10} y_5 \\ -y_4 + \alpha_9 y_5 \end{pmatrix},$$

and

$$L_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -\phi_3 y_5 \end{pmatrix}.$$

4) Taking the derivatives of the vectors L_1 and L_2 (eqs. (2-9) through (2-11)), we obtain

$$L_{11} = \begin{bmatrix} 0 & 0 & \frac{-\alpha_2 y_1}{\alpha_4} & \frac{-\alpha_1 y_1}{\alpha_4} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \frac{-\alpha_2 y_1}{\alpha_4} & 0 & 0 & 0 & 0 \\ \frac{-\alpha_1 y_1}{\alpha_4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Again, since all α's except α_{11} are constant, the L_{12} matrix reduces to a column vector,

5)

$$L_{12} = \begin{pmatrix} 0 \\ 0 \\ -y_5 \\ 0 \\ 0 \end{pmatrix},$$

and the L_{22} matrix reduces to the scalar 0.

Since the code was originally designed for a simple problem where the variables were previously determined, the implementation of this problem required the addition of two subroutines to calculate the corresponding y_j's for the function $L_1 = \nabla_1 R + \nabla_1 N y = 0$ and the corresponding ϕ's from $N(\phi, \alpha) = 0$. The insertion of flags to detect a sign change in $\dot{\lambda}$ for location of limit points was also included.

Furthermore, while implementing this problem into the code, a subroutine was included to extend the code to handle the case when the Jacobian, $F'(u)$, became singular. However, this was never applicable to this problem therefore it remains a part of the code which has not yet been tested.

3.2 – Results for the BWR Test Problem

After implementing a flag to signal a sign change in $\dot{\lambda}$ (i.e., to detect a limit point), the code confirmed the two equilibrium points and the first bifurcation point for the BWR model described by equations (3-6) through (3-10). Beginning

with the point $\alpha_{11} = \alpha_1 \geq -0.01300$ and $\{\phi_1, \phi_2, \phi_3, \phi_4, \phi_5\} = 0.0$, the homotopy procedure located the stable equilibrium point

$$A = \begin{cases} \phi_1 = .1, \\ \phi_2 = 175.0, \\ \phi_3 = 0, \\ \phi_4 = 0, \\ \phi_5 = 0, \end{cases}$$

and also the first bifurcation point at $k_0 = \alpha_1 = -0.01320$. See Figure 2. Since this method does not yet have the capability to locate the higher order bifurcations (i.e., $k_1, k_2, \ldots$), as α_1 was increased to a value greater than approximately -0.020, the unstable equilibrium point was located at

$$B = \begin{cases} \phi_1 = -1.0, \\ \phi_2 = -1750.0, \\ \phi_3 = -118.5, \\ \phi_4 = -17.83\alpha_1, \\ \phi_5 = -75.83\alpha_1 + 3.093 \times 10^{-3}. \end{cases}$$

The results obtained for the equilibrium points were in exact agreement with the ones reported by March–Leuba et al, but the value for k_0 differs by $+0.00002$.

3.3 – Constrained Optimization Test Problem

The second problem is a simple optimization problem[7] used solely to test the active constraint option implemented in the code. This version required the addition of flags to detect the active constraints and subroutines to handle the extra calculations needed for the active constraints. An important element for implementing this problem into the code was the need for a flag to signal whether the problem was one of maximization or minimization.

The problem is to maximize,

$$f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2 + 40x_1 + 20x_2 - 3000, \qquad (3\text{-}16)$$

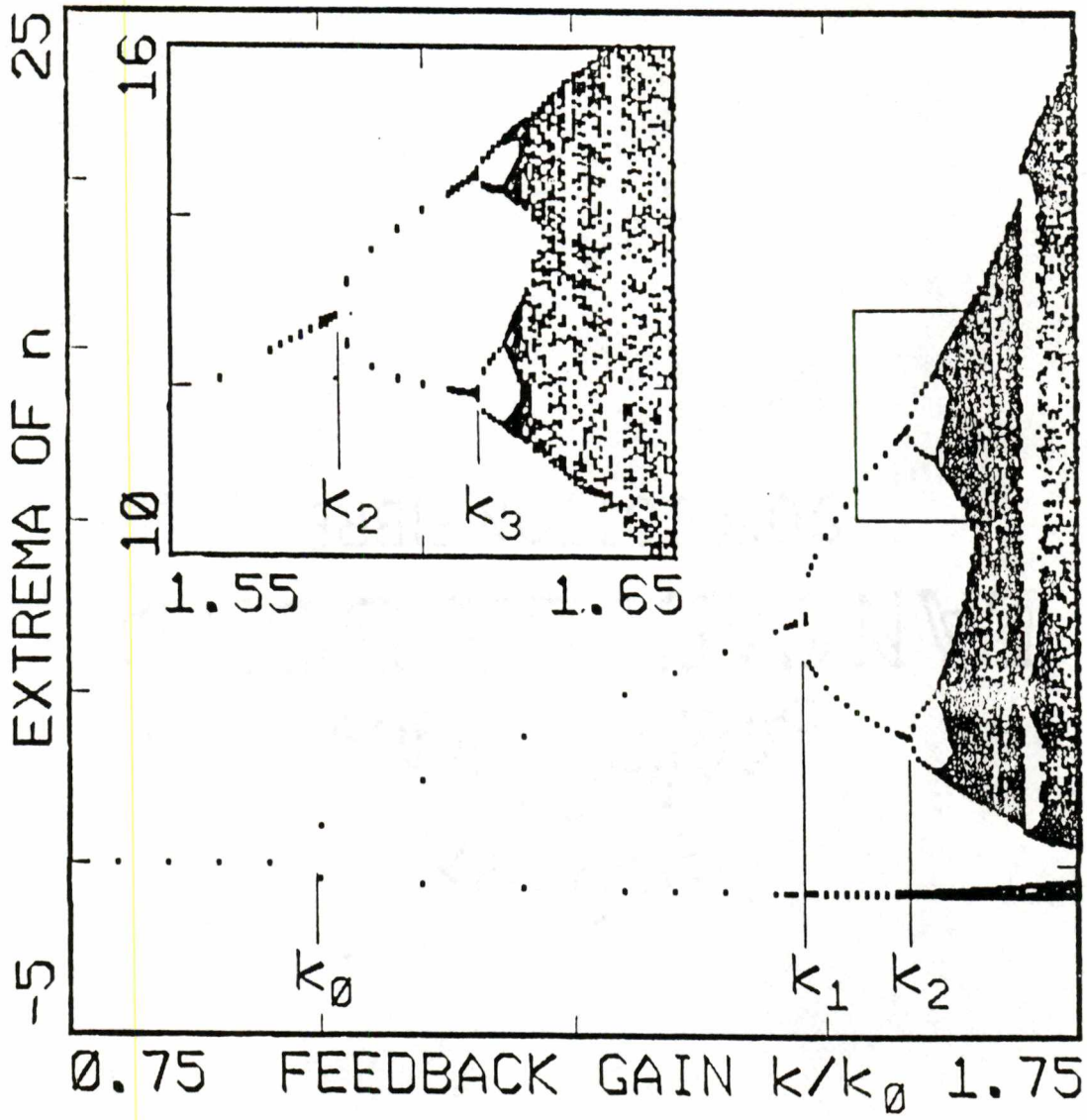


Figure 2. Bifurcation diagram as reported by March–Leuba et al.[3]

subject to

$$g_1(x_1, x_2, x_3) = x_1 - 50 \geq 0, \tag{3-17}$$

$$g_2(x_1, x_2, x_3) = x_1 + x_2 - 100 \geq 0, \tag{3-18}$$

$$g_3(x_1, x_2, x_3) = x_1 + x_2 + x_3 - 150 \geq 0. \tag{3-19}$$

Due to the algebraic nature of this problem, there are no state variables, ϕ, or equations, $N(\phi, \alpha)$, and by substituting α's for the x's and defining $f(x_1, x_2, x_3)$ $= R(\alpha)$, we obtain

$$F'(u) = \begin{pmatrix} L_{22} & \nabla g_I & \nabla g_J \\ 2Z_I(\nabla g_I)^T & 0 & 0 \\ 0 & 0 & 2C_J \end{pmatrix},$$

and

$$F(u) = \begin{pmatrix} L_2 \\ K_I(g_I(\alpha), z) \\ K_J(g_J(\alpha), z) \end{pmatrix}.$$

Now to define the components of $F'(u)$ and $F(u)$.

1) In $F(u)$, the function $L_2 = \nabla_2 R + (\nabla g_I)z$ is defined as the vector quantity,

$$L_2 = \begin{pmatrix} 2\alpha_1 + 40 + z_1 + z_2 + z_3 \\ 2\alpha_2 + 20 + z_2 + z_3 \\ 2\alpha_3 + z_3 \end{pmatrix},$$

and

2) the $K_{I,J}(g(\alpha), z)$ vectors, defined by equation (2-6), depend on the number of active and/or inactive constraints contained in the problem.

3) In $F'(u)$, define

$$L_{22} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{pmatrix},$$

4) ∇g is divided into ∇g_J and ∇g_I as,

$$\nabla g = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

and

5) the matrices C_J and Z_I, defined by equations (2-12) and (2-13), depend on the number of active and/or inactive constraints.

Note that if the problem is one of maximization and the constraints are active ($g(\alpha) \geq 0$) then $z_j < 0$. On the other hand, if the problem is a minimization one with active constraints ($g(\alpha) \geq 0$) then $z_j > 0$.[7]

3.4 – Results for the Constrained Optimization Test Problem

For the constrained optimization problem, a number of flags were installed to mark the active constraints as they appeared. Since the code was originally designed for inactive constraints, the flags which previously had signalled the calculations of variables that pertained to only inactive constaints now allow for the possibility of active constraints as well. A subroutine was also added so that by beginning with any combination of α_1, α_2, and α_3 subject to the constraints stated in equations (3-17) through (3-19), the code calculates the corresponding z_j's. The homotopy procedure then yielded the solution point,

$$\alpha_1 = 50, \qquad z_1 = -20,$$

$$\alpha_2 = 50, \qquad z_2 = -20,$$

$$\alpha_3 = 50, \qquad z_3 = -100.$$

for the maximization of equation (3-16).

The code also calculated a solution for the inactive constraint case where $g_j(\alpha) < 0$ and $z_j = 0$ which started from the point $\{\alpha_1, \alpha_2, \alpha_3\} = 0$ to be

$$\alpha_1 = -20,$$

$$\alpha_2 = -10,$$

$$\alpha_3 = 0.$$

CHAPTER IV

CONCLUSIONS AND RECOMMENDATIONS

The methodology presented in Reference 1 was designed to determine the global solutions of a nonlinear system, without linearizations, accurately and efficiently. Therefore, the incorporation of this methodology into a computational code created a valuable tool for practically analyzing nonlinear systems.

The objectives accomplished in this thesis included extending this code to locate limit points and to handle inactive or active constrained optimization problems. Verification of the method and the extended code was accomplished by using an unconstrained nonlinear BWR problem and a simple constrained optimization problem. The results obtained from the application of the test problems presented in this thesis verified the new method of analysis as well as the capabilities of the analysis code. It has been found that this methodology provides a considerable improvement in CPU time over the traditional methods for locating simple bifurcation points. The results also revealed that the enhancements made during these applications actually broadened the method's scope of analysis.

Another area in which this formalism could make a major contribution, is in the area of optimal control theory where nonlinear equations, such as a cost functions, depend on weight parameters. This code could be adapted to determine those weight parameters.

Recommendations for future work include the following:

(a) extending the code to identify the higher order bifurcations,

(b) finding a better step–length (σ) estimation procedure,

(c) testing the method incorporated to solve the Jacobian matrix when $F'(u)$ becomes singular,

(d) extending the method to perform a more comprehensive analysis of larger scale systems,

and

(e) extending the scope of the method to determine the exchange of stability between the solution branches.

LIST OF REFERENCES

LIST OF REFERENCES

1. D. G. Cacuci, "A Methodology for Global Analysis and Optimization of Nonlinear Systems", Nucl. Sci. Eng., submitted.

2. D. G. Cacuci, "Deterministic Sensitivity Theory for Nonlinear Systems: The Forward and Adjoint Methods", ORNL-TM-10058, Martin Marietta Energy Systems, Engineering Physics and Mathematics Div., Oak Ridge Natl. Lab., 1986.

3. J. March–Leuba, D. G. Cacuci, and R. B. Perez, "Nonlinear Dynamics and Stability of Boiling Water Reactors: Parts 1 and 2", Nucl. Sci. Eng., Vol. 93, p. 111, 1986.

4. H. B. Keller "Numerical Solution of Bifurcation and Nonlinear Eigenvalue Problems", *Applications of Bifurcation Theory*, P. H. Rabinowitz, Ed., p. 359, Academic Press, New York, 1977.

5. H. B. Keller, "Numerical Solution of Two Point Boundary Value Problems" *Regional Conference Series in Applied Mathematics*, Vol. 24, SIAM, Philadelphia, PA, 1976.

6. H. B. Keller, "Numerical Methods for Two–Point Boundary–Value Problem", Blaisdell Publishing Co., Waltham, Mass., 1968.

7. S. S. Rao, "Optimization Theory and Applications", Wiley Eastern Limited, New Delhi, India, 1979.

8. W. C. Rheinboldt, "Methods for Solving Systems of Nonlinear Equations", *Regional Conference Series in Applied Mathematics*, No. 14, SIAM, Philadelphia, PA, 1974.

APPENDIX

FORTRAN SOURCE CODE

This appendix contains the fortran listing of the code used in this thesis. Various changes can and have been made to accommodate the particular problem.

```
C *** MAIN
        DIMENSION D(10000)
        LIMIT = 10000
        N7 = 7
        N6 = 3
        IS = 3
        M  = 3
        N3 = 3
        N6N6 = N6*N6
        MM2 = M * M
        MIS = M * IS
         IFD = M + IS
         MPREC = 2
         NRF = IFD
         KA1 = NRF + 1
        L1 = 1
C *** ALP
        L2 = L1 + N3 * MPREC
C *** Z
        L3 = L2 + N6 * MPREC
C *** G
        L4 = L3 + N6 * MPREC
C *** FL2
        L5 = L4 + N3 * MPREC
C *** VL2
        L6 = L5 + IFD * MPREC
C *** JS
        L7 = L6 + N6
C *** ISA
        L8 = L7 + N6
C *** A
        L9 = L8 + IFD*IFD*MPREC
C *** B
        L10 = L9 + KA1 * KA1 * MPREC
C *** AD
        L11 = L10 + MPREC * KA1*KA1
C *** BD
        L12 = L11 + MPREC * KA1
C *** S
        L13 = L12 + KA1 * MPREC
C *** WORK
            IF((L13/2)*2.EQ.L13) L13 = L13 +1
        L14 = L13 + MPREC*(KA1*(KA1+3))
C *** E11(M,M)
        L15 = L14 + MM2 * MPREC
C *** E12(M,M)
        L16 = L15 + MM2 * MPREC
C *** E13(M,IS)
```

```
        L17 = L16 + MIS * MPREC
C *** E32(IS,M)
        L18 = L17 + MIS * MPREC
C *** E33(IS,IS)
        L19 = L18 + IS * IS * MPREC
C *** ISA(N6)
        L20 = L19 + N6
C *** OPGI(IS,N6)
        L21 = L20 + N6N6 * MPREC
C *** PG(IS,N6)
        L22 = L21 + N3 * N6  * MPREC
C *** OPGJ(IS,N6)
        L23 = L22 + N6N6 * MPREC
C *** A1(     )
        L24 = L23 + IFD * N6 * MPREC
C *** JSI(N6)
        L25 = L24 + N6 * MPREC
C *** VL21
        L26 = L25 + IFD * MPREC
C *** C(N6,N6)
        L27 = L26 + N6 * N6 * MPREC
C *** CI(N6,N6)
        L28 = L27 + N6 * N6 * MPREC
C *** GME(IS)
        L29 = L28 + IS  * MPREC
C *** GVEC
        L30 = L29 + IFD * MPREC
C *** FL2I(IS)
        L31 = L30 + IS  * MPREC
C *** IPVT
        L32 = L31 + KA1
C *** FLU
            IF((L32/2)*2.EQ.L32) L32 = L32 +1
        L33 = L32 + IFD * MPREC * IFD
C *** RES
        L34 = L33 + IFD * MPREC
C *** DX
        L35 = L34 + IFD * MPREC
C *** XR
        L36 = L35 + IFD * MPREC
C *** UL
        L37 = L36 + IFD * MPREC
C *** FN
        L38 = L37 + M * MPREC
C *** FK
        L39 = L38 + N6  * MPREC
C *** FVEC
        L40 = L39 + NRF * MPREC
C *** FPF
```

```
        L41 = L40 + NRF * MPREC
C *** FPRIM
        L42 = L41 + NRF * NRF  * MPREC
C *** UDOT
        L43 = L42 + NRF * MPREC
C *** UU1
        L44 = L43 + NRF * MPREC
C *** UZERO
        L45 = L44 + NRF * MPREC
C *** UTER
        L46 = L45 + NRF * MPREC
C *** UDIF
        L47 = L46 + NRF * MPREC
C *** AVEC
        L48 = L47 + NRF * MPREC
C *** UDB
        L49 = L48 + NRF * MPREC
C *** BVH
        L50 = L49 + NRF * MPREC
C *** BVA
        L51 = L50 + NRF * MPREC
C *** FVIT
        L52 = L51 + NRF * MPREC
C *** DUIT
        L53 = L52 + NRF * MPREC
C *** UZIT
        L54 = L53 + NRF * MPREC
C *** EQUIL
            IF((L54/2)*2.EQ.L54) L54 = L54 +1
        L55 = L54 + IFD * MPREC
C *** EALP
        L56 = L55 + N3  * MPREC
C *** GAP
        L57 = L56 + N3  * MPREC
C *** BM
        L58 = L57 + N6  * MPREC
C *** UDOTD
        L59 = L58 + NRF * MPREC
C *** ASEEK
        L60 = L59 + KA1 * KA1  * MPREC
C *** FVNT
        L61 = L60 + NRF * MPREC
C *** VL1
        L62 = L61 + M * MPREC
C *** FV00
        L63 = L62 + NRF * MPREC
C *** ISPACE
        L64 = L63 + KA1
C *** UKEEP
```

```
      L65 = L64 + NRF * MPREC
      IF(L65.GT.LIMIT) STOP 66
      CALL BOSS(D(L1), D(L2), D(L3), D(L4), D(L5), D(L6), D(L7),
     *          D(L8), D(L9), D(L10), D(L11), D(L12), D(L13),
     *          D(L14), D(L15), D(L16), D(L17), D(L18), D(L19),
     *          D(L20), D(L21), D(L22), D(L23), D(L24), D(L25),
     *          D(L26), D(L27), D(L28), D(L29), D(L30),
     *          D(L31), D(L32), D(L33), D(L34), D(L35), D(L36),
     *          D(L37), D(L38), D(L39), D(L40), D(L41), D(L42),
     *          D(L43), D(L44), D(L45), D(L46), D(L47), D(L48),
     *          D(L49), D(L50), D(L51), D(L52), D(L53), D(L54),
     *          D(L55), D(L56), D(L57), D(L58), D(L59), D(L60),
     *          D(L61), D(L62), D(L63), D(L64), D(L58),
     *          N3, M, IS, N6, N7, IFD,NRF, KA1)
      RETURN
      END
      SUBROUTINE BOSS(ALP, Z, G, FL2, VL2, JS, ISA, A, B,
     *  AD, BD, S, WORK, E11, E12, E13, E32, E33,OPEN, OPGI,
     *  PG, OPGJ, A1, JSI, VL21, C, CI, GME, GVEC,
     *  FL2I,IPVT,FLU,RES,DX,XR,UL,FN,FK,FVEC,FPF,FPRIM,
     *  UDOT,UU1,UZERO,UTER,UDIF,AVEC,UDB,BVH,BVA,FVIT,
     *  DUIT,UZIT,EQUIL,EALP,GAP,BM,UDOTD,ASEEK,FVNT,
     *          VL1,FV00,ISPACE,UKEEP,BI,
     *          N3, M, IS, N6, N7,IFD,NRF,KA1)
      IMPLICIT REAL * 8 (A-H,O-Z)
        COMMON /TRACK1/ NPOINT, NUNU
      DIMENSION WORK(1), AD(IFD,1), BD(1), FLU(IFD,1), RES(1),
     *  UKEEP(1),XR(1), UL(1), EQUIL(1), DX(1),ISPACE(1)
      DIMENSION E11(M,1), E12(M,1), E13(M,1), E32(IS,1),
     *  E33(IS,1),FPRIM(NRF,1),OPGI(N6,1), PG(N3,1), OPGJ(N3,1),
     *  A1(IFD,1), JSI(1), VL21(1), C(N6,1), CI(N6,1), OPEN(1),
     *  GME(1), GVEC(1), FL2I(1), A(IFD,1), UDOTD(1),ASEEK(KA1,1)
      DIMENSION FL2(1), VL2(1), Z(1), B(IFD,1), S(1), G(1),
     *  ALP(1), JS(1), ISA(1), IPVT(1),  FN(1), FK(1),FVNT(1),
     *  FVEC(1), FPF(1), UDOT(1), UU1(1,1), UZERO(1), VL1(1),
     *  UTER(1), UDIF(1), AVEC(1), UDB(1), BVH(1), BVA(1), BI(1),
     *  FVIT(1), DUIT(1),UZIT(1),EALP(1),GAP(1),BM(1),FV00(1)
      DIMENSION BSIG(2), BXI(2), SIG3(3), ZI(3,3)
C *** CGRK, GREEK C
      READ(5,21) EPS,DELTA, XNRUN,SIGN9,X9OUT
        IRUN = XNRUN + 0.001
        I9OUT = X9OUT + 0.0001
 21   FORMAT(5E10.4)
      WRITE(10,22) EPS,DELTA,IRUN,I9OUT,SIGN9
      WRITE(6,22) EPS,DELTA,IRUN,I9OUT,SIGN9
 22   FORMAT(1H ,'EPSILON',1P,E12.4,2X,'DELTA',1P,E12.4/
     *  1H ,'IRUN, I9OUT, SIGN9',2I5,1P,E15.5)
C *** NITER, ITERATION COUNT
C ****    I9OUT= 0
```

```
         NITER = 0
        NPOINT = 0
         IFIRST = 0
         XLZERO = 1.0
         CGRK = 1.0
         BXI(1) = 1.0
C ****   SCANT, FLAG, = 0 OFF, GT 0 ON
         SCANT = 0.0
C *** FDATA - DEFINES ALP, Y1, Y2, P1, P2, Z
        CALL FDATA(ALP, Y1,Y2,Y3,Y4,Y5,P1, P2,P3,P4,P5, Z, N6)
        WRITE(6,100)ALP(1),ALP(2),ALP(3),Z(1),Z(2),Z(3)
  100   FORMAT(1H,'ALP(1),ALP(2),ALP(3)',1P,3E12.4/1H,'Z(1),Z(2),Z(3)',
     *     1P,3E12.4)
C *** DEFINE U VECTOR WITH INITIAL VALUES, UKEEP, KEEP FOREVER
        CALL DUVEC(UKEEP, P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,ALP, Z)
C *** DEFINE U VECTOR WITH INITIAL VALUES, 1ST ITERATE INITIAL.
        CALL DUVEC(UZERO, P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,ALP, Z)
C *** DEFINE VECTOR (0,0,L2)
        CALL DFL2(ALP,Z,Y1,Y2,Y5,P1,P3,FL2)
        CALL DFL2I(FL2,VL2,M,IS)
C *** FPREP, UP FRONT DATA SETUP FOR CALC. UP TO ASEEK MATRIX
         CALL FPREP(G, ALP, ISA, JS, IC, JC, UTER, OPGI, ZI,
     *        Z,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,E11,E12,E13,E32,E33,FL2I,
     *        A, VL2, S, AD, BD, IFD, IPVT, FLU, RES, DX, XR, UL,
     *        EQUIL, C, CI, FK, BM, PG, OPGJ, EALP, FN, VL1,IFD1,
     *        IS, N6, M, N3, NPOINT, NUNU, I9OUT,IFLAG,B,A1,BI,VL21)
         WRITE(6,20)DELTA
   20    FORMAT(1H,'***DELTA',1P,E12.4)
C **** FUDOT, CALC. THE INITIAL UDOT
         CALL FUDOT(UDOT, FPRIM, A, C, OPGJ, FVEC, VL1, ISA,JS,
     *          FN,FL2,FK,ALP,Z,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,FPF,S,
     *             FV00, UU1, ASEEK, AD, WORK, FVNT, FVIT, SIGMA,IFD1,
     *            ISPACE, UDOTD, XLAM, DELTA, SIGN9,IC,JC,OPGI,ZI,
     *            KA1, M, IS, N3, NRF,IFD, N6,I9OUT,NPOINT,NITER,IFLAG)
C *** DUTER, CALC INITIAL UTER AND XI
         CALL DUTER(UTER, CGRK, XLZERO, SIGMA, XLAM,IS,N6,
     *                    UDOT, UKEEP, NRF, I9OUT,ISA)
         CALL UTUPD(UTER,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,ALP,Z,N3,N6)
    3    CONTINUE
        NPOINT = NPOINT + 1
        IF(NPOINT.GT.IRUN) STOP 33
        NITER = 0
        WRITE(6,12) NPOINT
   12   FORMAT(1H0,'$$$$LOOKING FOR POINT NUMBER$$$$',I8)
    2    CONTINUE
         NITER = NITER + 1
          NUNU = NITER - 1
         WRITE(6,11) NUNU
   11    FORMAT(1H0,'----  ITERATION NUMBER, NU=',I8,'  ---')
```

```
          WRITE(6,16) (ALP(I),I=1,N3)
  16      FORMAT(1H ,'THE BASIC DATA, ALPHA'/
     *     1H ,1P,7E12.4)
C *** FPREP, UP FRONT DATA SETUP FOR CALC. UP TO ASEEK MATRIX
          CALL FPREP(G, ALP, ISA, JS, IC, JC, UTER, OPGI, ZI,
     *        Z,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,E11,E12,E13,E32,E33,FL2I,
     *        A, VL2, S, AD, BD, IFD, IPVT, FLU, RES, DX, XR, UL,
     *        EQUIL, C, CI, FK, BM, PG, OPGJ, EALP, FN, VL1,IFD1,
     *        IS, N6, M, N3, NPOINT, NUNU, I9OUT,IFLAG,B,A1,BI,VL21)

C *** DGSB, SETUP THE GVECTOR  AND SOLVE LINEAR SYSTEM
          CALL DGSB(BVH,GAP,ALP,Z,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,FL2I,
     *                  GME, EALP, VL1, FN, FK, GVEC, AD, BD, IPVT,
     *                  FLU, RES, DX, XR, UL, EQUIL, UKEEP, A, FL2,
     *                  CGRK, M, IS, I9OUT, IFD, NPOINT, NITER, KA1,
     *                  N3,IFLAG,B,BI,IFD1)
C ***      I9OUT = 0
C ***     CALL SHIFT2(BSIG,BXI,SIGMA,CGRK,SCANT,NITER,0)
C *** CALC. A PART OF THE M SCALAR ON PAGE J
          CALL PARTM(FMMM, VL1, FL2,ALP,S,FN,FK,FVEC,FPF,FPRIM,
     *    IPVT,UDOTD,A,C,  UDOT,UU1,UZERO,UTER,UDIF,AVEC,DUIT,UZIT,
     *      BVH,BVA, CGRK, M, IS, N6, IFD, NRF, KA1, ASEEK, OPGJ,
     *      G,Z,AD,WORK,FVNT,FVIT,XLZERO,IFIRST,NITER,
     *      Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,DSIT,XLAM,DELTA,FV00,
     *      GAP, FL2I, GME, GVEC, BD, RES, DX, XR,UL,FLU,
     *      EALP, EQUIL, ISPACE, SIGN9, I9OUT, SCANT, SIGMA,NPOINT)
          CALL SHIFT2(BSIG,BXI,SIGMA,CGRK,SCANT,NITER,0)
          CALL  EDITJ(DUIT,DSIT,ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,CGRK,
     *      NUNU,NPOINT,N3,Z)
          CALL  EDITK(FMMM,XLAM,SIGMA,CGRK,XLZERO,
     *      NUNU,NPOINT)
C *** CALCULATE NEXT ITERATE
C *** TEST PROBLEM ONLY
          CALL CNEXTI(DUIT,DSIT,ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,Z,
     *         CGRK,N6,N3,ISA)
           IF(SCANT.GT.0.0)
     *    CALL SHIFT2(BSIG,BXI,SIGMA,CGRK,SCANT,NITER,1)
C *** TEST CONVERGENCE
          CALL CCONV(DUIT, DSIT, EPS, ISIG, NRF, NPOINT, NUNU)
            IF(SCANT.GT.0.0) CALL SECANT(SIGMA,BXI,BSIG,SIG3)
C *** ISIG = 0, INDICATES NO CONVERGENCE
          IF(ISIG.EQ.0) GO TO 2
          WRITE(6,15)
  15      FORMAT(1H ,'POP THE CORK, WE HAVE CONVERGENCE')
          IF(SCANT.LT.1.0)
     *    CALL SHIFT2(BSIG,BXI,SIGMA,CGRK,SCANT,NITER,1)
C ***   WE HAVE CONVERGENGE WITH SECANT IN EFFECT
C ***  AROOT, SETS UP TO LOOK FOR ANOTHER ROOT
          IF(SCANT.LT.1.0) GO TO 28
```

```
          SCANT = 0.0
          CALL AROOT(ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,Z,CGRK,SIG3,N6,ISA)
          GO TO 4
   28     CONTINUE




C *** SETUP CONDITIONS TO LOOK FOR ANOTHER POINT
C *** BPAGEK, PREPARE TO LOCATE A NEW POINT
           CALL BPAGEK(GCDOT,UTDOT,UTER,AVEC,UU1,ALP,UDOT,CGRK,
     *     Z,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,XLAM,XLZERO,DSIT,NRF,
     *     N3,N6,SCANT)
    4       CONTINUE
C *** DEFINE U VECTOR WITH CURRENT VALUES, UZERO
         CALL DUVEC(UZERO,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,ALP,Z)
C *** GET SETUP TO CALC. THINGS AS FUNCTION OF UZERO
C *** FPZIP, UP FRONT DATA SETUP FOR FUZIP
          CALL FPZIP(G, ALP, ISA, JS, IC, JC, UTER, OPGI,
     *         Z,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,E11,E12,E13,E32,E33,FL2I,
     *         A, VL2, S, AD, BD, IFD, IPVT, FLU, RES, DX, XR, UL,
     *         EQUIL, C, CI, FK, BM, PG, OPGJ, EALP, FN, VL1,
     *         IS, N6, M, N3, IFD, NPOINT, NUNU, I9OUT, IFLAG)
           IF(I9OUT.GT.0) CALL PRMAT(VL1,M,1)
C *** FUZIP, CALC. THINGS AS FUNCTION OF UZERO
          CALL FUZIP(FMMM, VL1, FL2,ALP,S,FN,FK,FVEC,FPF,FPRIM,
     *   IPVT,UDOTD,A,C, UDOT,UU1,UZERO,UTER,UDIF,AVEC,DUIT,UZIT,
     *    BVH, BVA,  CGRK, M, IS, N6, IFD, NRF, KA1,ASEEK,OPGJ,OPGI,
     *    G, Z, AD, WORK, FVNT, FVIT, XLZERO, IFIRST,NITER,IFD1,JS,
     *    Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,DSIT,XLAM,DELTA,FV00,ISA,
     *    GAP, FL2I, GME, GVEC, BD, RES, DX, XR, UL, FLU,IC,JC,ZI,
     *    EALP, EQUIL,ISPACE,SIGN9,I9OUT,SCANT,SIGMA,NPOINT,IFLAG)
         IF (SCANT.GT.1.0) GO TO 713
         IF (DABS(XLZERO) .LT. 0.05) GO TO 713
         SIGMA =0.05
   713    CONTINUE
C *** DUTER, CALC NEW POINT UTER AND XI
          CALL DUTER(UTER, CGRK, XLZERO, SIGMA, XLAM,IS,N6,
     *                    UDOT, UZERO, NRF, I9OUT,ISA)
          CALL UTUPD(UTER,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,ALP,Z,N3,N6)
          WRITE(6,8)BSIG(1),BSIG(2),BXI(1),BXI(2)
    8      FORMAT(1H,'BSIG',2E12.4,'BXI',2E12.4)
           IF(SCANT.GT.0.0) CALL SECANT(SIGMA,BXI,BSIG,SIG3)
             IF(1.EQ.1) GO TO 3
          RETURN
          END
```

```
      SUBROUTINE FILLZ(A, NR, NC)
    IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION A(NR,NC)
      DO 1 J = 1,NC
      DO 1 I = 1,NR
      A(I,J) = 0.0
1     CONTINUE
      RETURN
      END
      SUBROUTINE FILLA(A, E, IP, JP, NR, NC, IFD)
    IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION A(IFD,1), E(NR,NC)
      DO 1 J = 1,NC
      DO 1 I = 1,NR
      A(I+IP,J+JP) = E(I,J)
1     CONTINUE
      RETURN
      END
      SUBROUTINE FILLAT(A, E, IP, JP, NR, NC, IFD)
    IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION A(IFD,1), E(NC,1)
      DO 1 J = 1,NC
      DO 1 I = 1,NR
      A(I+IP,J+JP) = E(J,I)
1     CONTINUE
      RETURN
      END
      SUBROUTINE DL11(E,M,Y1)
    IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION E(M,1)
      CALL FILLZ(E,M,M)
      E(3,1) = (.540 *Y1)
      E(4,1) = (-3750*Y1)
      E(1,3) = (.540 *Y1)
      E(1,4) = (-3750*Y1)
      RETURN
      END
      SUBROUTINE DON1(P1,P2,P3,P4,E,M,ALP)
    IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION E(M,1),ALP(1)
      DO 1 I=1,5
       DO 1 J=1,5
1       E(I,J) = 0.0D0
      E(1,1) =(-1*(0.15*P4-.0000252*P3-.0056)/.00004)
      E(2,1) = -0.08
      E(3,1) = (-1*(P1+1)*(-.0000252)/.00004)
      E(4,1) = (-1*(P1+1)*0.15/.00004)
      E(1,2) = -.0056/.00004
      E(2,2) = 0.08
```

```
      E(1,3) = -25.044
      E(3,3) = 0.2325
      E(5,4) = -1.0
      E(3,5) = -ALP(1)
      E(4,5) = 6.8166
      E(5,5) = 2.2494
      RETURN
      END
      SUBROUTINE DL12(E,M,I,Y5)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION E(M,I)
      CALL FILLZ(E,M,I)
      E(3,1) = -Y5
      RETURN
      END
      SUBROUTINE DON2(ALP,E,I,M,P3)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION E(I,M), ALP(1)
     DO 5 J=1,5
5        E(1,J)=0.0D0
     E(1,5) = -P3
      RETURN
      END
      SUBROUTINE DL22(E,I,M)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION E(M,1)
      E(1,1)=2.0D0
      E(2,2)=2.0D0
      E(3,3)=2.0D0
C ***  WRITE(6,11) Y1,Y2,ALP(1),ALP(2),ALP(3),C1,C2,C3,C4,C5,
C    *   ((E(J1,J2),J2=1,I),J1=1,I)
 11   FORMAT(1H ,'DEBUG DL22',1P,E12.4/(1H ,1P,8E12.4))


      RETURN
      END
      SUBROUTINE DFA(A,ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,
     *  E11, E12, E13, E32, E33, IS, M, M3, IFD)
     IMPLICIT REAL * 8 (A-H,O-Z)
C *** DEFINES THE A MATRIX
      DIMENSION ALP(M3), E11(M,1), E12(M,1), E13(M,1),
     *  E32(IS,M), E33(IS,1), A(IFD,1)
C     CALL DL11(E11,M,Y1)
C     CALL DON1(P1,P2,P3,P4,E12,M,ALP)
C     CALL DL12(E13,M,IS,Y5)
C     CALL DON2(ALP,E32,IS,M,P3)
      CALL DL22(E33,IS,M)
      CALL FILLZ(A,IFD,IFD)
      M2 = M + M
```

```
C          CALL FILLA (A, E11, 0, 0, M, M, IFD)
C          CALL FILLA (A, E12, 0, M, M, M, IFD)
C          CALL FILLA (A, E13, 0,M2, M, IS, IFD)
C          CALL FILLAT(A, E12, M, 0, M, M, IFD)
C          CALL FILLAT(A, E32, M,M2, M,IS, IFD)
C          CALL FILLAT(A, E13,M2, 0,IS, M, IFD)
C          CALL FILLA (A, E32,M2, M,IS, M, IFD)
           CALL FILLA (A, E33,0,0,IS,IS, IS)
           RETURN
           END
C *** DEFINE THE VECTOR (0,0,L2) FOR SOLUTION OF G.1
           SUBROUTINE DFL2I(FL2,VL2,M,IS)
          IMPLICIT REAL * 8 (A-H,O-Z)
           DIMENSION VL2(1), FL2(1)
           M3 = M + M
           DO 1 I = 1,IS
           VL2(I) = 0.0
  1        CONTINUE
           M31 = M3 + 1
           M32 = M3 + IS
           DO 2 I = 1,IS
           VL2(I) = FL2(I)
   2       CONTINUE
           RETURN
           END
C *** SYSTEM OF EQUATIONS WITH IMSL LEQT2F
           SUBROUTINE SOLV1(AA,BB,S,AD,BD,IFD,IPVT,FLU,RES,DX,XR,UL,EQUIL,
     *                    NPOINT,NUNU,M,ISS,IC,FN,VL1,XLAM,UDOT,FVEC,GME)
          IMPLICIT REAL * 8 (A-H,O-Z)
           DIMENSION AD(ISS,1),BD(1),RES(1),DX(1),XR(1),UL(1),VL1(1)
           DIMENSION EQUIL(1),FLU(ISS,1),UDOT(1),FVEC(1)
           DIMENSION AA(ISS,1),BB(ISS),S(1),IPVT(1),FN(1),GME(1)
           DIMENSION AS(6,6),UT(6,6), WK(22)
           DO 12 K = 1,ISS
  12         S(K) = 0.0
           DO 1 J = 1,ISS
           BD(J) = BB(J)
  1        CONTINUE
           DO 20 J = 1,ISS
            DO 21 I = 1,ISS
              AD(I,J) = AA(I,J)
              AS(I,J) = AA(I,J)
  21        CONTINUE
  20       CONTINUE
           CALL FILLZ(UT,ISS,ISS)
           ID = 10
           CALL LUDATF(AD,FLU,ISS, ISS, ID, D1, D2, IPVT, EQUIL,
     *                 WA,IER)
           WRITE(6,4) IER,IFD, WA
```

```
    4     FORMAT(1H,'EXITED FROM LUDATF',2I5,1P,E12.4)
C         IF (IER.NE.129) GO TO 8
C         CALL SINGU(AS,IFD,XR,GME,M,ISS,IC,FN,VL1,FVEC,UDOT,XLAM,FK,
C    *          VL2)
C         GO TO 7
C   8     CONTINUE

          CALL LUELMF(FLU,BD, IPVT, ISS, ISS, XR)
          CALL LUREFF(AD,BD,FLU,IPVT,ISS,ISS,XR,ID,RES,DX,IER1)

          WRITE(6,2) IER
    2     FORMAT(1H ,'PAGE G SOLUTION, IER',I6)
    7     CONTINUE
          DO 3 J = 1,ISS
          S(J) = XR(J)
    3     CONTINUE
          WRITE(6,5) (S(I),I=1,ISS)
    5     FORMAT(1H0,'SOLUTION OF LINEAR SYSTEM INACTIVE,(A) SEE PAGE G'/
      *    (1H ,1P,11E12.4))
          WRITE(14,6) NPOINT,NUNU,IER,IER1,WA,(S(I),I=1,ISS)
    6     FORMAT(1H ,'A',5I5/1H,11E12.4)
          RETURN
          END

              SUBROUTINE SINGU(AS,IFD,XR,GME,M,IS,IC,
      *                        FN,VL1,FVEC,UDOT,XLAM,FK,VL2)
          IMPLICIT REAL *8 (A-H,O-Z)
          DIMENSION AS(IFD,IFD),UT(11,11),V(11,11),S(11),WK(22)
          DIMENSION Q(11,11),ST(11),GV(11),B(11),QU(11,11)
          DIMENSION XP(11),PHY(11),CT(11),VQU(11,11),XR(IFD)
          DIMENSION BA(11,11),STA(11),XS(11),GME(1)
          DIMENSION XS1(11),FVEC(IFD),UDOT(IFD)
          DIMENSION VL1(1),FN(1),FK(1),VL2(1)
          MD=M+M+IS
          DO 1 L=1,IFD
           DO 2 J=1,IFD
            UT(L,J)=0.0D0
            QU(L,J)=0.0D0
            BA(L,J)=AS(L,J)
            VQU(L,J)=0.0D0
            Q(L,J)=0.0D0
    2      CONTINUE
    1     CONTINUE
          DO 3 L=1,IFD
    3     UT(L,L)=1.0D0
          DO 15 L=1,IFD
            B(L)=-FVEC(L)
            CT(L)=UDOT(L)
   15     CONTINUE
```

```
      CALL LSVDF(BA,IFD,IFD,IFD,UT,IFD,IFD,S,WK,IER)
      DO 5 L=1,IFD
       DO 5 J=1,IFD
 5      V(L,J)=BA(L,J)
                    CALL PRMAT(V,IFD,IFD)
                    CALL PRMAT(UT,IFD,IFD)
                          WRITE(6,200)(S(L),L=1,IFD)
                          WRITE(6,101) IER
101                       FORMAT(I6)
      DO 65 L=1,IFD
       IF (DABS(S(L)).LT. 1.0E-8) S(L)=0.0D0
 65    CONTINUE

C ***  RIGHT HAND SIDE ***
      CALL GVECS(GV,VL1,FN,GME,M,IS,VL2,FK)
      GAMMA = 0.0D0
                         WRITE(6,200)(GV(K),K=1,IFD)

      DO 6 L=1,IFD
       IF(DABS(UT(IFD,L)).LT.1.0E-6) UT(IFD,L)=0.0D0
        ST(L)=UT(IFD,L)
       IF(DABS(V(L,IFD)).LT.1.0E-6) V(L,IFD)=0.0D0
        PHY(L)=V(L,IFD)
        XP(L)=0.0D0
        XR(L)=0.0D0
 6    CONTINUE
       CP=0.0D0
       CX=0.0D0
        D=XLAM
      STB=0.0D0
      STG=0.0D0
      DO 7 L=1,IFD
        STB=ST(L)*B(L)+STB
        STG=ST(L)*GV(L)+STG
 7    CONTINUE
       DO 8 L=1,IFD
        IF (DABS(STB).LT.1.0E-8) GO TO 12
        XS1(L)=B(L)*(1.0D0/STB)+XS1(L)
        GO TO 8
12    XS1(L)= 0.0D0
 8    CONTINUE
      DO 9 K=1,IFD
 9     XS(K)=GV(K)-(XS1(K)*STG)
      DO 50 K=1,IFD
       IF (S(K).EQ.0.0) GO TO 50
       Q(K,K)=1.0D0/S(K)
 50   CONTINUE
      CALL MMULT(QU,Q,UT,IFD,IFD,IFD)
      CALL MMULT(VQU,V,QU,IFD,IFD,IFD)
```

```
      DO 31 L=1,IFD
       DO 32 J=1,IFD
        XP(L)=VQU(L,J)*XS(J)+XP(L)
   32   CONTINUE
   31  CONTINUE
                          WRITE(6,200)(XP(L),L=1,IFD)
  200                     FORMAT(1H,11D11.3,1X)
       IF (DABS(STB).LT. 1.0E-8) GO TO 20
       SI=(1.0D0/STB)*STG
       GO TO 25
   20  SI=0.0D0
   25  CONTINUE
       DO 70 L=1,IFD
        CP=CT(L)*PHY(L)+CP
        CX=CT(L)*XP(L)+CX
   70  CONTINUE
        DSI=D*SI
       IF(DABS(CP).LT. 1.0E-8) GO TO 81
       Z=(1.0D0/CP)*(GAMMA-DSI-CX)
       GO TO 84
   81  Z=0.0D0
   84  CONTINUE
      DO 82 L=1,IFD
   82  XR(L)=XP(L)+PHY(L)*Z
                          WRITE(6,200)(XR(L),L=1,IFD)
       RETURN
       END


       SUBROUTINE GOPI(ISA,IC,PG,OPGI,M6,M3)
C *** DEFINE G OPERATOR FOR ACTIVE COLUMNS
C *** ISA, ACTIVE COLUMN NUMBERS
C *** IC,  THE NUMBER OF ACTIVE COLUMNS
C *** PARTIAL DERIVATIVES, SEE PAGE D
      IMPLICIT REAL * 8 (A-H,O-Z)
       DIMENSION ISA(1), PG(M3,1), OPGI(M3,M6)
       JJ = 0
       CALL FILLZ(OPGI,M3,M3)
       DO 1 I = 1,IC
       II = ISA(I)
       DO 2 J = 1,M3
       OPGI(J,I) = PG(J,II)
   2   CONTINUE
   1   CONTINUE
       RETURN
       END
       SUBROUTINE GOPJ(JS,JC,PG,OPGJ,M6,M3)
C *** DEFINE G OPERATOR FOR INACTIVE COLUMNS
C *** JS, INACTIVE COLUMN NUMBERS
```

```
C *** JC,  THE NUMBER OF INACTIVE COLUMNS
C *** PARTIAL DERIVATIVES, SEE PAGE D
        IMPLICIT REAL * 8 (A-H,O-Z)
         DIMENSION JS(1), PG(M6,1), OPGJ(M6,M6)
         CALL FILLZ(OPGJ,M6,M6)
         DO 1 I = 1,JC
         II = JS(I)
         DO 2 J = 1,M6
         OPGJ(J,I) = PG(J,II)
   2     CONTINUE
   1     CONTINUE
         CALL PRMAT(OPGJ,M6,M6)
         RETURN
         END
C *** DEFINE MATRIX A1(U1), SEE PAGE D
         SUBROUTINE DFA1(OPGI,A1,IC,M,IS,MD)
        IMPLICIT REAL * 8 (A-H,O-Z)
         DIMENSION OPGI(IS,1),A1(MD,1)
C *** MD = M+M+IS
         CALL FILLZ(A1, MD, IC)
         CALL FILLA(A1, OPGI, 0, 0, IS,IC, IS)
         CALL PRMAT(A1,MD,IC)
         RETURN
         END
C *** DEFINE THE MATRIX B TO BE USED IN SOLUTION PAGE G
         SUBROUTINE DFB(B, A, A1, IS, IC, M, IFD, IFD1, IFD2     )
        IMPLICIT REAL * 8 (A-H,O-Z)
         DIMENSION B(IFD1,1), A(IFD2,1) ,A1(IFD2,1)
C *** IFD = M + M + IS + M + M +IC
      IFD1 = IS +IC
C *** IFD2 = IS
         NC = IFD2
C *** FILL B WITH ZEROES
         CALL FILLZ(B,IFD1,IFD1)
C *** PUT A INTO B
         CALL FILLA(B,A,0,0,IFD2,IFD2,IFD1)
C *** PUT A1 INTO B
         CALL FILLA(B,A1,0, IFD2, IFD2, IC, IFD1)
C *** PUT A1 TRANSPOSE INTO B
         CALL FILLAT(B,A1,IFD2,0,IC,IFD2,IFD1)
         RETURN
         END
C *** DEFINE THE VECTOR (0,0,L2,0) FOR THE SOLUTION OF G.2
         SUBROUTINE DFL2A(VL2, VL21, IFD1, IFD)
        IMPLICIT REAL * 8 (A-H,O-Z)
         DIMENSION VL2(IFD1), VL21(IFD)
         LS = IFD1 + 1
         DO 1 I = 1,IFD1
         VL21(I) = VL2(I)
```

```
          VL21(I+3) = 0.0
    1     CONTINUE
          RETURN
          END
C *** CAINT- CALCULATES CONSTRAINT DATA
          SUBROUTINE CAINT(G, ISA, JS, IC, JC, N6, IFD1)
         IMPLICIT REAL * 8 (A-H,O-Z)
          DIMENSION G(1), ISA(1), JS(1)
          IC = 0
          JC = 0
          DO 3 I=1,N6
          ISA(I)=0
          JS(I)=0
   3      CONTINUE
          DO 1 I = 1,N6
          IF(G(I).LT.0.0) GO TO 2
C *** ACTIVE
          IC = IC + 1
          ISA(IC) = I
          GO TO 1
  2       CONTINUE
C *** INACTIVE
          JC = JC + 1
          JS(JC) = I
  1       CONTINUE
          IFD1 = N6 + IC
          WRITE(6,5)JC,IC
  5       FORMAT(1H,'# INACTIVE CONST',I5,4X,'# ACTIVE CONST',I5)
          RETURN
          END
C *** CALG - CALCULATES G (CONSTRAINT)
          SUBROUTINE CALG(G,ALP,M3,M6)
         IMPLICIT REAL * 8 (A-H,O-Z)
          DIMENSION G(M6), ALP(M3)
C **  M3 = 3
C **  M6 = 2
          DO 1 I=1,M6
  1        G(I)=0.0D0

          G(1) = ALP(1) - 50.0
          G(2) = ALP(1) + ALP(2) - 100.0
          G(3) = ALP(1) + ALP(2) + ALP(3) - 150.0
          RETURN
          END
C *** DFC - DEFINE MATRIX C
          SUBROUTINE DFC(C, G, JS, JC, N6)
         IMPLICIT REAL * 8 (A-H,O-Z)
C *** IS C A 3X3, SEE PAGE C
          DIMENSION G(1), C(N6,1), JS(1)
```

```
      CALL FILLZ(C,N6,N6)
      DO 1 I = 1,JC
      J = JS(I)
      C(J,J) = G(J)
 1    CONTINUE
      RETURN
      END
C *** DFK - DEFINE MATRIX K
C *** NOTE - IS IT (3X1)
      SUBROUTINE DFK(FK, G, Z, N6)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION FK(N6,1), G(1), Z(1) ,Z1(3), G1(3)
      DO 1 I = 1,N6
      Z1(I) = Z(I)
      G1(I) = G(I)
      IF (Z1(I) .LT. 0.0) Z1(I) = -Z1(I)
      IF (G1(I) .GT. 0.0) G1(I) = -G1(I)
      FK(I,1)=(G1(I) + Z1(I))**2+G1(I)*DABS(G1(I))-Z1(I)*DABS(Z1(I))
 1    CONTINUE
      RETURN
      END
C *** GY - DEFINE GY, REALLY N, SEE PAGE H
      SUBROUTINE GY(FN, P1, P2,P3,P4,P5, ALP,NUNU)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION FN(1), ALP(1),TT(4)
      FN(1) =(-P1*(0.15D0*P4-.0000252D0*P3-.0056D0)/.00004D0-0.08D0*
     *     P2-(0.15D0*P4-.0000252D0*P3)/.00004D0)
      FN(2) =(-P1*0.0056D0/.00004D0 +0.08D0*P2)
      FN(3) = (-25.04D0*(P1-0.1D0)+0.2325D0*P3)
      FN(4) = -P5
      FN(5)=(2.2494*P5+6.8166*P4-ALP(1)*P3)
      DO 5 I=1,5
        IF (DABS(FN(I)).LT.1.0E-10) FN(I)=0.0
 5    CONTINUE
      RETURN
      END
C *** GALP, G(ALP)
      SUBROUTINE GALP(GAP,ALP,Z,Y1,Y2,Y5,P1,P3,FL2I,FL2,CGRK,M,IS)
        IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION ALP(1), Z(1), FL2I(1), FL2(1), GAP(1)
      CALL DFL2(ALP, Z, Y1, Y2,Y5, P1,P3, FL2I)
      DO 1 I = 1,IS
      GAP(I) = FL2I(I) - FL2(I)*CGRK
 1    CONTINUE
      RETURN
      END
C *** MMULT - MATRIX MULTIPLICATION - NUMBER OF COLUMNS OF THE FIRST
C ***         MUST = NUMBER OF ROWS OF THE SECOND
C ***         (NR X NC) (NC X NC2) = (NR X NC2)
```

```
      SUBROUTINE MMULT(C,A,B,NR,NC,NC2)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION C(NR,1), A(NR,1), B(NR,1)
      DO 1 J = 1,NC2
      DO 1 I =1,NR
      SUM = 0.0
      DO 2 K = 1,NC
      SUM = SUM + A(I,K) * B(K,J)
2     CONTINUE
      C(I,J) = SUM
1     CONTINUE
      RETURN
      END
C *** CINV - INVERT THE C MATRIX, WHICH IS A DIAGONAL MATRIX
      SUBROUTINE CINV(CI, C, N6)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION C(N6,1), CI(N6,1)
      DO 1 I = 1,N6
        IF(C(I,I).EQ.0.0) GO TO 1
      CI(I,I) = 1.0/C(I,I)
1     CONTINUE
      RETURN
      END
C *** MSCLE - MATRIX MULTIPLICATION X SCALAR
C
C
      SUBROUTINE MSCLE(A,SCALE,NR,NC)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION A(NR,1)
      DO 1 J = 1,NC
      DO 1 I =1,NR
      A(I,J) = A(I,J) * SCALE
1     CONTINUE
      RETURN
      END
C *** GEALP, CALC. GALP - EALP
      SUBROUTINE GEALP(GME, G, E, IS)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION GME(1), G(1), E(1)
       DO 1 I = 1,IS
       GME(I) = G(I) - E(I)
1      CONTINUE
       RETURN
       END
C *** GVECS, DEFINE THE G VECTOR
C *** GVEC IS VECTOR OF LENGTH 2M+IS, COMPOSED OF
C *** VL2, A VECTOR OF LENGTH M
C *** FK,  A VECTOR OF LENGTH M
      SUBROUTINE GVECS(GVEC, VL1, FN, GMS, M, IS, BI)
```

```
      IMPLICIT REAL * 8 (A-H,O-Z)
       DIMENSION GVEC(1), VL1(1), FN(1), GMS(1), BI(1)
       CALL FILLZ(GVEC,M+IS,1)
       DO 1 I = 1,M
       GVEC(I) = GMS(I)
       GVEC(I+M) = BI(I)
 1     CONTINUE
       RETURN
       END
C *** FILLV, FILLS A VECTOR
       SUBROUTINE FILLV(VEC, VIN, LS, LC)
      IMPLICIT REAL * 8 (A-H,O-Z)
       DIMENSION VEC(1), VIN(1)
C *** LS, STARTING POINTER
C *** LC, NO. OF COMPONENTS IN VIN
       DO 1 I = 1,LC
       VEC(I+LS) = VIN(I)
 1     CONTINUE
       RETURN
       END
C *** DVL1, DEFINE L1 VECTOR OF LENGTH M
       SUBROUTINE DVL1(VL1, Y1, Y2,Y3,Y4,Y5, P1, P2,P3,P4,ALP)
      IMPLICIT REAL * 8 (A-H,O-Z)
       DIMENSION VL1(1),TT(4),ALP(1)
       T1=(0.15D0*P4-.0000252D0*P3-.0056D0)/.00004D0
       VL1(1) = (-T1*Y1-140.0D0*Y2-25.044D0*Y3)
       VL1(2) = (-0.08D0*Y1+0.08D0*Y2)
       VL1(3) =(1.0D0-(P1+1)*(-.540D0)*Y1+.2325D0*Y3-ALP(1)*Y5)
       VL1(4) = (-1*(P1+1)*3750.0D0*Y1+6.8166D0*Y5)
       VL1(5) = (-Y4+2.2494D0*Y5)
       DO 1 I=1,5
        IF(DABS(VL1(I)).LT. 1.0E-10) VL1(I)=0.0
 1     CONTINUE
         WRITE(6,11) VL1(1),VL1(2),VL1(3),VL1(4),
    *          VL1(5)
11        FORMAT(1H ,'CHECK VL1',1P,5E13.5)
       RETURN
       END
C *** MNORM - MATRIX NORMAL
        SUBROUTINE MNORM(A,XN,NR,NC)
      IMPLICIT REAL * 8 (A-H,O-Z)
        DIMENSION A(NR,1)
        DIMENSION  AD(20,20)
        SUM = 0.0
        DO 1 J = 1,NC
        DO 1 I =1,NR
        SUM = SUM + A(I,J) **2
 1      CONTINUE
C ***   XN = DDSQRT(SUM)
```

```
      DO 3 J = 1,20
      DO 3 I = 1,20
    3 AD(I,J) = 0.0
      DO 4 J = 1,NC
      DO 4 I = 1,NR
    4 AD(I,J) =A(I,J)
        CALL VNRMF2(AD,NR,20,XNN)
        XN = XNN
      RETURN
      END
C *** PRMAT, PRINT A MATRIX
      SUBROUTINE PRMAT(A,NR,NC)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION A(NR,1)
       WRITE(6,3)
    3  FORMAT(1H0)
      DO 2 I = 1,NR
      WRITE(6,1) (A(I,J),J=1,NC)
    1 FORMAT(1H ,1P,13E10.2)
    2 CONTINUE
      WRITE(6,4)
    4 FORMAT(1H ,'----------------------------')
      RETURN
      END
      SUBROUTINE FDATA(ALP, Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5, Z, N6)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION ALP(1), Z(1),X(5),W(3)
      ALP(1) = 49.0D0
      ALP(2) = 49.0D0
      ALP(3) = 49.0D0
C     P1 = -7.00D0
C     P2 = -50.0D0
C     P3 = -10.50D0
C     P4 = 0.22D0
C     P5 = 0.0D0
C      CALL YSOLV(P1,P2,P3,P4,ALP,M,X)
C     Y1=X(1)
C     Y2=X(2)
C     Y3=X(3)
C     Y4=X(4)
C     Y5=X(5)
       CALL ZSOLV(ALP,N6,W)
      Z(1) = W(1)
      Z(2) = W(2)
      Z(3) = W(3)
      RETURN
      END
C *** CINV - INVERT THE C MATRIX
      SUBROUTINE XINV(AD, BD, WORK, C, CI, N6)
```

```
                                8 (A-H,O-Z)
                               .), WORK(1), BD(N6,1)
                               ,1), CI(N6,1)


                               J)


                               , N6, N6, BD, IDGT, WORK, IER)

                               INVERSE MESSAGE',I6)


              CI(I,J) = BD(I,J)
    3         CONTINUE
              RETURN
              END
C*** DUVEC, DEFINES THE U VECTOR
              SUBROUTINE DUVEC(UZIT,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,ALP,Z)
             IMPLICIT REAL * 8 (A-H,O-Z)
              DIMENSION UZIT(1), ALP(1), Z(1)
              UZIT(1)  = ALP(1)
              UZIT(2) = ALP(2)
              UZIT(3)  = ALP(3)
              DO 1 I = 1,3
              UZIT(I+3) = Z(I)
    1         CONTINUE
              RETURN
              END
C *** PARTM, CALC. A PART OF THE M SCALAR ON PAGE J
              SUBROUTINE PARTM(FMMM, VL1, FL2,ALP,S,FN,FK,FVEC,FPF,FPRIM,
         *  IPVT,UDOTD,A,C,    UDOT,UU1,UZERO,UTER,UDIF,AVEC,DUIT,UZIT,
         *   BVH, BVA,  CGRK, M, IS, N6, IFD, NRF, KA1,ASEEK,OPGI,
         *   G, Z, AD, WORK, FVNT, FVIT, XLZERO, IFIRST,NITER,
         *   Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,DSIT,XLAM,DELTA,FV00,
         *   GAP, FL2I, GME, GVEC, BD, RES, DX, XR, UL, FLU,
         *   EALP, EQUIL,ISPACE,SIGN9,I9OUT,SCANT,SIGMA,NPOINT)
C *** DEFINE F AND F PRIME, SEE PAGE J
C *** DAN WILL PROVIDE MORE INFORMATION
            IMPLICIT REAL * 8 (A-H,O-Z)
          DIMENSION AD(KA1,1), WORK(1)
C *** TEMPORARY DIMENSIONS FOR PARTM
            DIMENSION FVEC(1), FN(1), FK(1), FPF(1), UDOT(1), S(1),
         *  UU1(1,1), UZERO(1), UTER(1), UDIF(1), AVEC(1),UDOTD(1),
         *  BVH(1), BVA(1), DUIT(1), UZIT(1),FPRIM(NRF,1),OPGI(IS,1),
         *  VL1(1), FL2(1),A(IFD,1),C(N6,1),IPVT(1),ASEEK(KA1,1),
         *  ALP(1), G(1), Z(1), FVNT(1), FVIT(1), FV00(1),ISPACE(1),
         *  GAP(1), FL2I(1), GME(1), GVEC(1), BD(1), RES(1),
         *  DX(1), XR(1), UL(1), FLU(1), EALP(1), EQUIL(1)
```

```
          DIMENSION ALPF(3)
C *** END OF TEMP.
C ***     I9OUT= 0
           N3 = IS
            IF(I9OUT.GT.0) WRITE(6,104) NITER
 104         FORMAT(1H ,'IN PARTM, NITER=',I7)
         IF(I9OUT.GT.0) WRITE(6,212)
  212    FORMAT(1H ,' THE UTER VECTOR, V VECTOR ITERATION DEP. ')
           IF(I9OUT.GT.0) CALL PRVEC(UTER,NRF,  1)
          T1 = XLAM * (CGRK - XLZERO) -SIGMA
C*** CALC. U CURRENT - UZERO
          DO 3 I = 1,NRF
          UDIF(I) = UTER(I) - UZERO(I)
   3      CONTINUE
C ***       I9OUT = 0
         IF(I9OUT.GT.0) WRITE(6,204)
 204     FORMAT(1H ,'----U ITERATION  - U ZERO----')
          IF(I9OUT.GT.0) CALL PRVEC(UDIF,NRF,1)
         IF(I9OUT.GT.0) WRITE(6,205)
 205     FORMAT(1H ,'----U ITERATION ----')
          IF(I9OUT.GT.0) CALL PRVEC(UTER,NRF,1)
         IF(I9OUT.GT.0) WRITE(6,206)
 206     FORMAT(1H ,'----U ZERO --')
          IF(I9OUT.GT.0) CALL PRVEC(UZERO,NRF,1)
C *** CALC. PRODUCT OF UU1 X UDIF
          CALL MMULT(T2, UU1, UDIF, 1, NRF, 1)
C ***  DEFINE M
          FMMM = T2 + T1
C *** DEFINE A VECTOR
          DO 4 I = 1,NRF
          AVEC(I) = 0.0
   4      CONTINUE
          CALL FILLV(AVEC, S, 0,  IFD)
         IF(I9OUT.GT.0) WRITE(6,207)
 207     FORMAT(1H ,'----A  VECTOR AS USED TO FORM UDOT TRANSPOSE X A')
C *** PRODUCT UDOT TRANSPOSE X BVH
          IF(I9OUT.GT.0) CALL PRVEC(AVEC,NRF,1)
          CALL MMULT(UDB, UU1, BVH, 1,NRF,1)
C *** PRODUCT UDOT TRANSPOSE X AVEC
          CALL MMULT(UDA, UU1,AVEC, 1, NRF, 1)
         IF(I9OUT.GT.0) WRITE(6,208)
 208     FORMAT(1H ,'----B  VECTOR AS USED TO FORM UDOT TRANSPOSE X B')
          IF(I9OUT.GT.0) CALL PRVEC(BVH ,NRF,1)
C *** SEE TOP OF PAGE J
          DSIT = (UDB - FMMM)/(UDA + XLAM)
C *** EQ J.2
          DO 5 I = 1,NRF
          DUIT(I) = DSIT * AVEC(I) - BVH(I)
   5      CONTINUE
```

```
      IF(I9OUT.GT.0) WRITE(6,209)
 209  FORMAT(1H ,'----DELXI * A - B, SEE TO OF PAGE J ---')
       IF(I9OUT.GT.0) CALL PRVEC(DUIT,NRF,1)
       IF(I9OUT.GT.0)
     *  WRITE(6,102) XLAM,UU3,FMMM,T2,T1,XN,UDB,UDA,DSIT,UU2,CGRK,
     *               XLZERO,SIGMA
 102  FORMAT(1H ,'END OF PARTM',2X,'LAMBDA DOT',1P,E15.6,
     *  2X,'UDOT TRANS * U ZERO + LAMBDA DOT',1P,E15.6/1H ,
     *  'M',1P,E15.6,2X,'TERM 1 OF M',1P,E15.6,2X,'TERM 2 OF M',
     *   1P,E15.6,2X,'NORM SEE DEF. OF LAMBDA DOT',1P,E15.6/1H ,
     *   'UDOT TRANSPOSE * B',1P,E15.6,2X,'UDOT TRANSPOSE * A',
     *   1P,E15.6,2X,'DELXI',1P,E15.6,/1H ,
     *   'UDOT TRANSPOSE * U ZERO'
     *   ,1P,E15.6,'  XI',1P,E15.6/1H ,'LAMBDA ZERO',1P,E15.6,
     *   2X,'SIGMA',1P,E15.6)
C
       IFIRST = 9
C
C ***   I9OUT = 0
      RETURN
      END
C *** DFPRIM, DEFINES THE F PRIME MATRIX
      SUBROUTINE DFPRIM(FPRIM, A,C,KPD,IFD, N6, OPGJ, M,IS,IC,JC,
     *          OPGI,ZI,IFLAG,IFD1)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION FPRIM(KPD,1), A(IS,1), C(N6,1), OPGJ(N6,1)
      DIMENSION OPGIT(3,3),ZI(3,3),ZIT(3,3),OPGI(N6,1)
      IF (IFLAG .EQ. 0) IFD1=N6
      CALL FILLZ(FPRIM, KPD, KPD)
      CALL FILLA(FPRIM, A, 0, 0, IS, IS, KPD)
      CALL FILLA(FPRIM, OPGI,0, N6, IS, IC,KPD)
      CALL FILLA(FPRIM, OPGJ, 0, IFD1, IS, JC, KPD)
      IF (IFLAG .EQ. 0) GO TO 4
      SCALE=2.0
      CALL MSCLE(ZI,SCALE,IC,N6)
      CALL FILLAT(OPGIT,OPGI,0,0,N6,N6,N6)
      CALL PRMAT(OPGIT,N6,N6)
      CALL PRMAT(ZI,N6,N6)
      CALL MMULT(ZIT,ZI,OPGIT,N6,N6,N6)
      CALL PRMAT(ZIT,N6,N6)
      CALL FILLA(FPRIM,ZIT,N6,0,N6,N6,KPD)
  4   CONTINUE
       CALL PRMAT(C,N6,N6)
       J=0
       DO 1 I = 1,JC
  2     IF (J .GT. JC) GO TO 1
        J = J + 1
        IF (C(J,J) .EQ. 0.0) GO TO 2
        C(I,I) = 2.0 * C(J,J)
```

```
 1        CONTINUE
          CALL PRMAT(C,N6,N6)
          CALL FILLA(FPRIM, C, IFD1, IFD1, N6, JC, KPD)
          RETURN
          END
          SUBROUTINE RDATA(ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,Z,N6,NRF,
     *       DELTA,UDOT, UDOTD,ISA)
         IMPLICIT REAL * 8 (A-H,O-Z)
          DIMENSION ALP(1), Z(1), UDOT(1), UDOTD(1), ISA(1)
          DO 2 I = 1,NRF
          UDOTD(I) = DELTA * UDOT(I)
 2        CONTINUE
           WRITE(6,11) DELTA
 11         FORMAT(1H ,'DELTA',1P,E14.5/1H ,'UDOTD')
            CALL PRVEC(UDOTD,NRF,1)
          ALP(1) = ALP(1) + UDOTD(1)
          ALP(2) = ALP(2) + UDOTD(2)
          ALP(3) = ALP(3) + UDOTD(3)
          DO 3 I =1,N6
          J=ISA(I)
 3        Z(J) = Z(J) + UDOTD(I+3)
          WRITE(6,4) ALP(1),ALP(2),ALP(3),Z(1),Z(2),Z(3)
 4       FORMAT(1H ,'----BASIC DATA USED TO CALC(U0 + DELTA * UDOT), SEE
     *  CALC. OF SIGMA'/1H ,1P,11E12.4)
          RETURN
          END
C *** CSIGMA, CALCS. SIGMA
          SUBROUTINE CSIGMA(SIGMA,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,ALP,Z,
     *     FPRIM,ASEEK,FVEC,UDOT,AD,WORK,FVNT,FVIT,A,ISA, JS,
     *     DELTA, N3, IFD, NRF, N6,ISPACE,KA1, UDOTD, M, XLAM,FV00,
     *     I9OUT)
         IMPLICIT REAL * 8 (A-H,O-Z)
C         DIMENSION ALP(1), Z(1),  G(1), FN(1), FK(1),VL1(1),
C     *      FL2(1), ISA(1), JS(1)
C *** NOTE, DATA REMOVED FROM LIST AND FIXED DIMENSIO
C *** TEMPORARY TO KEEP ORG. DATA FROM BEING CHANGED
          DIMENSION   G(6), FN(5), FK(6),VL1(5),
     *      FL2(3), ALPH(3), ZH(6)
         DIMENSION FVEC(1),   UDOT(1),ISPACE(1),ALP(1), Z(1),FV00(1),
     *   ASEEK(KA1,1), FVIT(1), FVNT(1), FPRIM(NRF,1),UDOTD(1)
        DIMENSION  AD(KA1,1), WORK(1), A(IFD,1)
          COMMON /BUGS/ CNUM

          WRITE(6,20) DELTA
 20       FORMAT(1H,'***DELTA',1P,E12.4)
C ****    I9OUT= 0
C *** END OF TEMP.
C *** PRESERVE BASIC DATA
         DO 23 I = 1,N3
```

```
          ALPH(I) = ALP(I)
 23       CONTINUE
          DO 24 I = 1,N6
          ZH(I) = Z(I)
 24       CONTINUE
C *** UPDATE THE DATA, NEW F VECTOR
          IF(I9OUT.GT.0) WRITE(6,210)
 210      FORMAT(1H ,'----UDOT AGAIN IS IT OK????---')
              IF(I9OUT.GT.0) CALL PRVEC(UDOT,NRF,1)
          CALL RDATA(ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,Z,N6,NRF,DELTA,
     *         UDOT,UDOTD,ISA)


C *** FDATA, DEFINE DATA
C *** F VECTOR, FVEC IS 2M + IS + N6 IN LENGTH
C *** CALG - CALCULATES G (CONSTRAINT)
          CALL CALG(G,ALP,N3,N6)
C *** UPDATE DATA, REDEFINE F VECTOR
C *** DEFINE N, FN
C         CALL GY(FN,P1,P2,P3,P4,P5,ALP,NUNU)
C *** DEFINE THE L1 VECTOR, VL1, OF LENGTH M
C         CALL DVL1(VL1, Y1, Y2,Y3,Y4,Y5, P1, P2,P3,P4,ALP)
          CALL DFL2(ALP,Z,Y1,Y2,Y5,P1,P3,FL2)

C *** DEFINE MATRIX K, SEE PAGE H, K IS N6 X 1
          CALL DFK(FK, G, Z, N6)
C *** F VECTOR, FVEC IS 2M + IS + N6 IN LENGTH
C         CALL FILLV(FVIT, VL1, 0, M)
C         CALL FILLV(FVIT, FN,  M, M)
          CALL FILLV(FVIT, FL2, 0, N3)
          CALL FILLV(FVIT, FK, N3,N6)
          IF(I9OUT.GT.0) WRITE(6,211)
 211      FORMAT(1H ,'----F(U0 + DELTA * UDOT), SEE CALC. OF SIGMA')
                      IF(I9OUT.GT.0) CALL PRVEC(FVIT,NRF,1)
C *** DEFINE ASEEK
          CALL FILLZ(ASEEK, KA1, KA1)
C *** INSERT F
          CALL FILLA(ASEEK, FPRIM, 0, 0,NRF, NRF, KA1)
          DO 1 I = 1,NRF
          ASEEK(I,NRF+1) = - FV00(I)
          ASEEK(NRF+1,I) = UDOT(I)
  1       CONTINUE
          ASEEK(KA1,KA1)=XLAM
          IF(I9OUT.GT.0) WRITE(6,212)
 212      FORMAT(1H ,'----THE A SCRIPT MATRIX, SEE CALC. OF SIGMA')
          IF(I9OUT.GT.0) CALL PRMAT(ASEEK,KA1,KA1)
C *** RESTORE BASIC DATA
          DO 25 I = 1,N3
          ALP(I) = ALPH(I)
```

```
 25       CONTINUE
          DO 26 I = 1,N6
          Z(I) = ZH(I)
 26       CONTINUE
C ***
          CALL XDATA(ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,Z,N6,NRF,DELTA,
     *       UDOT,UDOTD,ISA)
C **
C *** FDATA, DEFINE DATA
C *** F VECTOR, FVEC IS 2M + IS + N6 IN LENGTH
C *** CALG - CALCULATES G (CONSTRAINT)
          CALL CALG(G,ALP,N3,N6)
C *** UPDATE DATA, REDEFINE F VECTOR
C *** DEFINE N, FN
C         CALL GY(FN,P1,P2,P3,P4,P5,ALP,NUNU)
C *** DEFINE THE L1 VECTOR, VL1, OF LENGTH M
C         CALL DVL1(VL1, Y1, Y2,Y3,Y4,Y5,P1,P2,P3,P4,ALP)
         CALL DFL2(ALP,Z,Y1,Y2,Y5,P1,P3,FL2)

C *** DEFINE MATRIX K, SEE PAGE H, K IS N6 X 1
         CALL DFK(FK, G, Z, N6)
C *** F VECTOR, FVEC IS 2M + IS + N6 IN LENGTH
C         CALL FILLV(FVNT, VL1, 0, M)
C         CALL FILLV(FVNT, FN,  M, M)
          CALL FILLV(FVNT, FL2, 0, N3)
          CALL FILLV(FVNT, FK, N3,N6)
          IF(I9OUT.GT.0) WRITE(6,213)
 213      FORMAT(1H ,'----F(U0 - DELTA * UDOT, SEE CALC. OF SIGMA')
                    IF(I9OUT.GT.0) CALL PRVEC(FVNT,NRF,1)
C*** GET MATRIX CONDITION NUMBER
          CALL MCOND(CNUM,ASEEK,AD,WORK,KA1,KA1,ISPACE)
          IF(I9OUT.GT.0) WRITE(6,413)
 413      FORMAT(1H ,'----F(U0) AGAIN IS IT OK,SEE CALC. OF SIGMA')
                    IF(I9OUT.GT.0) CALL PRVEC(FVEC,NRF,1)
          DO 2 I = 1,NRF
          T1 = DABS(FVIT(I))
          T2 = DABS(FVNT(I))
          T3 = DABS(FVEC(I))
          IF(T2.GT.T1) T1 = T2
          IF(T3.GT.T1) T1 = T3
          TN   =  FVIT(I) + FVNT(I) - 2.0*FVEC(I)
          IF(T1.NE.0.0) T2 = TN/T1
          IF(DABS(T2).LT.1.0E-12) TN = 0.0
          FVNT(I) = TN/(DELTA**2)
  2       CONTINUE
          IF(I9OUT.GT.0) WRITE(6,214)
 214      FORMAT(1H ,'----TAKE NORM OF THIS VECTOR,CALC. OF SIGMA')
            IF(I9OUT.GT.0) CALL PRVEC(FVNT,NRF,1)
C *** CALL MATRIX NORMAL
```

```
      CALL MNORM(FVNT, SNORM,NRF,1)
      T1 = SNORM * CNUM
      IF (T1.EQ.0.0) GO TO 31
      SIGMA = DSQRT(1.0D0/T1)
      GO TO 30
 31   SIGMA=0.004
 30   IF(I9OUT.GT.0) WRITE(6,11) CNUM,SNORM,T1,SIGMA
 11   FORMAT(1H0,'CONDITION NUMBER OF A SCRIPT MATRIX',1P,E12.4,2X,
     *  'NORM OF VECTOR',1P,E12.4/1H ,'PRODUCT',1P,E12.4,2X,'SIGMA',
     *   1P,E12.4)
C *** RESTORE BASIC DATA
      DO 27 I = 1,N3
      ALP(I) = ALPH(I)
 27   CONTINUE
      DO 28 I = 1,N6
      Z(I) = ZH(I)
 28   CONTINUE
      RETURN
      END
C *** MCOND, CALC. MATRIX CONDITION NUMBER
      SUBROUTINE MCOND(CNUM, ASEEK, A, Z, LDA, N, IPVT)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION A(LDA,1), Z(1)
      DIMENSION IPVT(1), ASEEK(LDA,1)
      DO 1 J = 1,N
      DO 1 I = 1,N
      A(I,J) = ASEEK(I,J)
  1   CONTINUE
      CALL DGECO(A, LDA, N, IPVT, RCOND, Z)
      IF(RCOND.EQ.0.0)GO TO 5
      CNUM = 1.0/RCOND
      GO TO 4
  5   CNUM=0.0
  4    WRITE(6,2) CNUM,RCOND
  2   FORMAT(1H ,'CONDITION NUMBER OF MATRIX',1P,2E14.5)
      RETURN
      END
C *** CNEXTI, CALC. NEXT ITERATE
       SUBROUTINE CNEXTI(DUIT,DSIT,ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,
     *      Z,CGRK, N6, N3,ISA)
     IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION DUIT(1), ALP(1), Z(1),ISA(1)
      DO 1 I = 1,N3
      ALP(I) = ALP(I) + DUIT(I)
 1    CONTINUE
      DO 2 I = 1,N6
      J=ISA(I)
      Z(J) = Z(J) + DUIT(I+3)
 2    CONTINUE
```

```
          CGRK = CGRK + DSIT
          RETURN
          END
C *** CCONV, TEST CONVERGENCE
          SUBROUTINE CCONV(DUIT, DSIT, EPS, ISIG, NRF, NPOINT, NUNU)
C *** ISIG = 9 INDICATES CONVERGENCE
        IMPLICIT REAL * 8 (A-H,O-Z)
          DIMENSION DUIT(1)
          ISIG = 0
          DO 1 I = 1,NRF
          IF(DABS(DUIT(I)).GE.EPS) GO TO 2
   1      CONTINUE
          IF(DABS(DSIT).LT.EPS) ISIG = 9
   2      CONTINUE
C *** COUNT AND DO NOT PRINT TRAILING 0 OF DUIT
          KZ = 0
          DO 5 I = 1,NRF
          IF(DUIT(NRF+1-I).NE.0.0) GO TO 6
          KZ = KZ + 1
   5      CONTINUE
   6      CONTINUE
          KZ = NRF - KZ
          WRITE(11,3)  NPOINT,NUNU,(DUIT(I),I=1,KZ),DSIT
   3      FORMAT(1H ,2I5,1P,8E12.4/(1H ,1P,8E12.4))
          WRITE(6,4) EPS, DSIT
   4      FORMAT(1H ,'EPSILON',1P,E12.4,2X,'DELTA XI',1P,E12.4)


          RETURN
          END
C *** DPG, DEFINE PG
          SUBROUTINE DPG(PG,N3)
        IMPLICIT REAL * 8 (A-H,O-Z)
          DIMENSION PG(N3,N3)
          PG(2,2) = 1.0D0
          PG(2,3) = 1.0D0
          PG(1,3) = 1.0D0
          PG(1,1) = 1.0D0
          PG(1,2) = 1.0D0
          PG(3,3) = 1.0D0
          RETURN
          END
C *** BPAGEK, PREPARE TO LOCATE A NEW POINT
          SUBROUTINE BPAGEK(GCDOT,UTDOT,UTER,AVEC,UU1,ALP,UDOT,CGRK,
     *    Z,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,XLAM,XLZERO,DSIT,NRF,N3,
     *    N6,SCANT)
        IMPLICIT REAL * 8 (A-H,O-Z)
          COMMON /TRACK1/ NPOINT, NUNU
          DIMENSION UTDOT(1), UTER(1), AVEC(1), ALP(1), Z(1),
```

```
     *    UU1(1,NRF),UDOT(1),CPRIA(11)
          CALL MNORM(AVEC, GCDOT, NRF, 1)
C ***    GREEK C DOT, SEE BOTTOM OF PAGE K
          GCDOT = 1.0/DSQRT(1.0D0 + GCDOT**2)
C *** CALC. UDOT
          DO 1 I = 1,NRF
          UDOT(I) = GCDOT * AVEC(I)
   1      CONTINUE
C *** UU1 IS UDOT0 TRANSPOSE
C ***
          CALL MMULT(UTD1, UU1, UDOT, 1, NRF, 1)
          UTD2 = UTD1 + XLAM * GCDOT
          WRITE(6,11) (UDOT(I),I=1,NRF)
  11      FORMAT(1H ,'UDOT',1P,8E12.4/1H ,1P,8E12.4)
          WRITE(6,12) UTD1,XLAM,GCDOT,UTD2
  12      FORMAT(1H ,1P,4E12.4)
          IF(UTD2.GT.0.0)GO TO 6
          GCDOT = -GCDOT
          DO 7 I = 1,NRF
          UDOT(I) = -UDOT(I)
   7      CONTINUE
   6      CONTINUE

          XLO=XLAM
          PXD=XLAM*GCDOT
          IF (PXD.GT.0.0) GO TO 40
          CALL MMULT(UDW,UU1,AVEC,1,NRF,1)
          CGRKP=1.0/(XLO+UDW)
          IF (PXD .LT. 0.0) SLDOT=9.0
          IF (PXD .LT. 0.0) WRITE(6,30) NPOINT,NUNU
  30      FORMAT(1H,'*** SIGN CHANGE LAMBDA DOT ***',2I6)
  40      CONTINUE
          WRITE(6,31) GCDOT,XLO
  31      FORMAT(1H,'NEW LAMBDA DOT',1P,E12.4,2X,'OLD LAMBDA DOT',1P,
     *       E12.4)

          XLAM = GCDOT
C *** END OF UDOT, XLAM SECTION
C *** TAKE CARE OF U ZERO AND XLAM 0
          DO 3 I = 1,N3
          ALP(I) = UTER(I)
   3      CONTINUE
          DO 2 I = 1,N6
          Z(I) = UTER(I+3)
   2      CONTINUE
  57        CONTINUE
          PD = XLZERO * CGRK
          IF(PD.LT.0.0) WRITE(16,4) NPOINT,NUNU
          IF(PD.LT.0.0) SCANT = 9.0
```

```
 4       FORMAT(1H ,'******* EUREKA ******* A SIGN CHANGE  ****',2I6)
         WRITE(16,5) CGRK, XLZERO, PD
 5       FORMAT(1H ,' NEW LAMBDA',1P,E12.4,2X,'OLD LAMBDA',1P,E12.4,
     *   2X,'THEIR PRODUCT',1P,E12.4)
         XLZERO = CGRK
         RETURN
         END
         SUBROUTINE XDATA(ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,Z,N6,NRF,
     *      DELTA, UDOT, UDOTD,ISA)
        IMPLICIT REAL * 8 (A-H,O-Z)
         DIMENSION ALP(1), Z(1), UDOT(1), UDOTD(1),ISA(1)
         DO 2 I = 1,NRF
         UDOTD(I) = DELTA * UDOT(I)
 2       CONTINUE
          WRITE(6,11) DELTA
 11        FORMAT(1H ,'DELTA',1P,E14.5/1H ,'UDOTD')
           CALL PRVEC(UDOTD,NRF,1)
         ALP(1) = ALP(1) - UDOTD(1)
         ALP(2) = ALP(2) - UDOTD(2)
         ALP(3) = ALP(3) - UDOTD(3)
         DO 3 I =1,N6
         J=ISA(I)
 3       Z(J) = Z(J) - UDOTD(I+3)
         WRITE(6,4) ALP(1),ALP(2),ALP(3),Z(1),Z(2),Z(3)
 4      FORMAT(1H ,'----BASIC DATA USED TO CALC(U0 - DELTA * UDOT), SEE
     *  CALC. OF SIGMA'/1H ,1P,11E12.4)
         RETURN
         END
C *** IDITJ
         SUBROUTINE EDITJ(DUIT,DSIT,ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,
     *      CGRK, NUNU, NPOINT, N3,Z)
         IMPLICIT REAL * 8 (A-H,O-Z)
         DIMENSION DUIT(1),ALP(1),Z(1)
         DATA NOLD /9999/
         IF(NUNU.LT.NOLD) WRITE(6,1)
 1       FORMAT(1H0)
         WRITE(10,2) NPOINT,NUNU,ALP(1),ALP(2),ALP(3),Z(1),Z(2),Z(3)
 2       FORMAT(1H ,2I5/1H,11E12.4)
         NOLD = NUNU
         RETURN
         END
         SUBROUTINE EDITK(FMMM,XLAM,SIGMA,CGRK,XLZERO,NUNU,NPOINT)
         IMPLICIT REAL * 8 (A-H,O-Z)
         DIMENSION DUIT(1),ALP(1)
         DATA NOLD /9999/
         IF(NUNU.LT.NOLD) WRITE(6,1)
 1       FORMAT(1H0)
         WRITE(12,2) NPOINT,NUNU,FMMM,XLAM,XLZERO,SIGMA,CGRK
 2       FORMAT(1H ,2I5,1P,5E14.5)
```

```
          NOLD = NUNU
          RETURN
          END
C *** SYSTEM OF EQUATIONS WITH IMSL LEQT2F
          SUBROUTINE SOLV2(A,B,S,AD,BD,IFD,IPVT,FLU,RES,DX,XR,UL,EQUIL,
     *                     NPOINT,NUN1,GME,FN,UDOT,VL1,M,IS,IC,XLAM,FVEC)
         IMPLICIT REAL * 8 (A-H,O-Z)
          DIMENSION AD(IS,1),BD(IS),RES(1),DX(1),XR(1),UL(1)
          DIMENSION EQUIL(1),FLU(IS,1),AS(11,11),VL1(1),FVEC(1)
          DIMENSION A(IS,1),B(1),S(1),IPVT(1),GME(1),FN(1),UDOT(1)
           NUNU = NUN1 - 1
          DO 12 K = 1,IS
  12        S(K) = 0.0
          DO 1 J = 1,IS
          BD(J)    = B(J)
          DO 10 I = 1,IS
          AD(I,J) = A(I,J)
  10      CONTINUE
   1      CONTINUE
          ID = 10
          CALL LUDATF(AD, FLU,IS, IS, ID, D1, D2, IPVT, EQUIL,
     *                WA,IER)
          WRITE(6,4) IER, WA
   4      FORMAT(1H ,'EXITED FROM LUDATF',I5,1P,E12.4)
C         IF(IER.NE.129) GO TO 8
C        CALL SINGU(A,AS,IFD,XR,GME,M,IS,IC,FN,VL1,FVEC,UDOT,XLAM,FK,
C    *           VL2)
C         GO TO 7
C  8      CONTINUE
          CALL LUELMF(FLU,BD, IPVT, IS, IS, XR)
          CALL LUREFF(AD, BD,FLU,IPVT,IS,IS,XR,ID,RES,DX,IER)

          WRITE(6,2) IER
   2      FORMAT(1H ,'PAGE G SOLUTION, IER',I6)
   7      CONTINUE
          DO 3 J = 1,IS
          S(J) = XR(J)
C          IF(DABS(S(J)).LT.1.0E-8) S(J)=0.0
   3      CONTINUE
          WRITE(6,5) (S(I),I=1,IS)
   5      FORMAT(1HO,'SOLUTION OF LINEAR SYSTEM INACTIVE,(B) SEE PAGE G'/
     *    (1H ,1P,11E12.4))
          WRITE(15,6) NPOINT,NUNU,IER,(S(I),I=1,IS)
   6      FORMAT(1H ,'B',3I5/1H,11E12.4)
          RETURN
          END
C *** PRMAT, PRINT A MATRIX
          SUBROUTINE PRVEC(A,NR,NC)
         IMPLICIT REAL * 8 (A-H,O-Z)
```

```
      DIMENSION A(1)
       WRITE(6,3)
 3      FORMAT(1HO)
       WRITE(6,1) (A(J),J=1,NR)
 1      FORMAT(1H ,1P,8E15.7/1H ,1P8E15.7)
       WRITE(6,4)
 4      FORMAT(1H ,'----------------------------')
       RETURN
       END
C ***    SUBROUTINE TO CALC. SECOND DERIVATIVE OF F
       SUBROUTINE DSIGMA(SIG2,UZERO,UDOT)
       IMPLICIT REAL * 8 (A-H,O-Z)
       COMMON /BUGS/ CNUM
       DIMENSION UZERO(1), UDOT(1), FV(2)
       SUM = 0.0
       DO 1 I =1,2
       SUM = SUM +  FV(I) **2
 1      CONTINUE
       SNOR = DSQRT(SUM)
       SIG2 = DSQRT(1.0/(SNOR*CNUM))
       RETURN
       END
C *** SHIFT, XI AND SIGMA
      SUBROUTINE SHIFT2(BSIG, BXI, SIGMA, CGRK, SCANT, NITER, KC)
      IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION BSIG(2), BXI(2)
      IF(KC.GT.0) GO TO 1
      IF(SCANT.GT.0.0) GO TO 3
      IF(NITER.GT.1) GO TO 2
 3      CONTINUE
C ***  SETUP DATA FOR THE SECANT METHOD
       BSIG(2) = BSIG(1)
       BSIG(1) = SIGMA
       GO TO 2
 1      CONTINUE
       BXI(2) = BXI(1)
       BXI(1) = CGRK
 2       CONTINUE
       RETURN
       END
C *** SECANT, CALCS. SIGMA BY SECANT METHOD
       SUBROUTINE SECANT(SIGMA, BXI, BSIG,SIG3)
       IMPLICIT REAL * 8 (A-H,O-Z)
       COMMON /TRACK1/ NPOINT, NUNU
       DIMENSION BXI(2), BSIG(2), SIG3(3)
       SIGMA = BSIG(1) - BXI(1)*(BSIG(1) - BSIG(2))/
     *    (BXI(1) - BXI(2))
        SIGK = BSIG(2)
C  5      CONTINUE
```

```
          WRITE(16,12) SIGMA,BSIG,BXI,NPOINT,NUNU
  12      FORMAT(1H ,'SECANT 1, SIGMA',1P,5E16.5,2I5)
          IF((BXI(1)*BXI(2)).GE.0.0) GO TO 2
          BIG = BSIG(1)
          IF(BSIG(2).GT.BIG) BIG = BSIG(2)
          SMALL = BSIG(1)
           SIGK = BSIG(2)
          WRITE(16,13) SIGMA,BIG,SMALL,NPOINT,NUNU
  13      FORMAT(1H ,'SECANT 2, SIGMA',1P,3E16.5,2I5)
          IF(BSIG(2).LT.SMALL) SMALL = BSIG(2)
          IF(SIGMA.LT.SMALL) GO TO 1
          IF(SIGMA.GT.BIG) GO TO 1
          GO TO 2
   1      CONTINUE
C *** REGULA FALSI
          SIGMA = (BXI(1) * BSIG(2) - BXI(2) * BSIG(1)) /
     *            (BXI(1) - BXI(2))
           WRITE(16,11) SIGMA,NPOINT,NUNU
   11      FORMAT(1H ,'REGULA FALSI, SIGMA',1P,E15.5,2I5)
   2       CONTINUE
           SIG3(1) = SIGMA
           SIG3(2) = BSIG(1)
           SIG3(3) = SIGK
          RETURN
          END
C *** DGSB, SETUP THE GVECTOR  AND SOLVE LINEAR SYSTEM
         SUBROUTINE DGSB(BVH,GAP,ALP,Z,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,
     *                 FL2I,GME, EALP, VL1, FN,FK,GVEC, AD, BD, IPVT,
     *                 FLU,RES, DX, XR, UL, EQUIL, UKEEP, A, FL2,
     *                 CGRK, M, IS, I9OUT, IFD, NPOINT, NITER, KA1,
     *                 N3,IFLAG,B,BI,IFD1)
        IMPLICIT REAL * 8 (A-H,O-Z)
         DIMENSION FN(1), BVH(1), GAP(1), ALP(1), Z(1),GME(1),EALP(1),
     *             VL1(1), GVEC(1), BD(1), IPVT(1), FLU(1), RES(1),
     *             DX(1), XR(1), UL(1), EQUIL(1), UKEEP(1), FL2I(1),
     *             AD(KA1,1), A(IFD,1), FL2(1), FK(1),BI(1),B(IFD1,1)
         IF(I9OUT.GT.0) WRITE(6,221) P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,
     *         ALP(1)
  221    FORMAT(1H ,'P1,P2,P3,P4,P5',1P,5D13.5 /1H ,'Y1,Y2,Y3,Y4,Y5',
     *          1P,5D13.5/1H ,'ALP(1)',1P,1D13.5)
   14       CONTINUE

C *** DEFINE GALP
         CALL GALP(GAP,ALP,Z,Y1,Y2,Y5,P1,P3,FL2I,FL2,CGRK, M, IS)
         IF(I9OUT.GT.0) WRITE(6,221) P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,
     *        ALP(1)
         IF(I9OUT.GT.0) WRITE(6,214)
   214   FORMAT(1H ,' THE FL2I VECTOR, COMP. OF G VECTOR       ')
           IF(I9OUT.GT.0) CALL PRVEC(FL2I, IS,  1)
```

```
      IF(I9OUT.GT.0) WRITE(6,215)
  215 FORMAT(1H ,' THE FL2  VECTOR, COMP. OF G VECTOR       ')
        IF(I9OUT.GT.0) CALL PRVEC(FL2,  IS,  1)
      IF(I9OUT.GT.0) WRITE(6,216)
  216 FORMAT(1H ,' THE GAP  VECTOR, COMP. OF G VECTOR       ')
        IF(I9OUT.GT.0) CALL PRVEC(GAP,  IS,  1)
C *** DEFINE GALP - EALP
      IF(I9OUT.GT.0) WRITE(6,217)
  217 FORMAT(1H ,' THE EALP VECTOR, COMP. OF G VECTOR       ')
        IF(I9OUT.GT.0) CALL PRVEC(EALP, IS,  1)
      CALL GEALP(GME, GAP, EALP, IS)
      IF(I9OUT.GT.0) WRITE(6,218)
  218 FORMAT(1H ,' THE GME  VECTOR, COMP. OF G VECTOR       ')
        IF(I9OUT.GT.0) CALL PRVEC(GME,  IS,  1)
C *** DEFINE THE L1 VECTOR, VL1, OF LENGTH M
C     CALL DVL1(VL1, Y1, Y2,Y3,Y4,Y5, P1, P2,P3,P4,ALP)
C     IF(I9OUT.GT.0) WRITE(6,219)
C 219 FORMAT(1H ,' THE VL1  VECTOR, COMP. OF G VECTOR       ')
C       IF(I9OUT.GT.0) CALL PRVEC(VL1,  M,   1)
C *** DEFINE N, FN
C     CALL GY(FN,P1,P2,P3,P4,P5,ALP,NUNU)
C     IF(I9OUT.GT.0) WRITE(6,220)
C 220 FORMAT(1H ,' THE FN   VECTOR, COMP. OF G VECTOR       ')
C       IF(I9OUT.GT.0) CALL PRVEC(FN,M,1)
      IF (IFLAG .NE. 3) GO TO 230
      CALL GVECS(GVEC,VL1,FN,GME,M,IS,BI)
       IF (I9OUT .GT. 0) WRITE(6,211)
       IF (I9OUT .GT. 0) CALL PRMAT(GVEC,IFD,1)
       IF (I9OUT .GT. 0) CALL PRMAT(B,IFD1,IFD1)
      CALL SOLV2(B,GVEC,BVH,AD,BD,IFD,IPVT,FLU,RES,
     *           DX,XR,UL,EQUIL,NPOINT,NITER,GME,FN,UDOT,VL1,M,
     *           IFD1,IC,XLAM,FVEC)
      GO TO 231
C *** DEFINE G VECTOR
  230  CALL GVECS(GVEC, VL1, FN, GME, M, IS,BI)
      IF(I9OUT.GT.0) WRITE(6,211)
  211 FORMAT(1H ,' THE G VECTOR, RIGHT SIDE OF LINEAR SYSTEM')
        IF(I9OUT.GT.0) CALL PRVEC(GVEC, IFD, 1)
C *** SOLVE LINEAR SYSTEM, SEE PAGE H, BVH IS SOLUTION VECTOR
       IF(I9OUT.GT.0) CALL PRMAT(A,IS,IS)
C *** SOLVE LINEAR SYSTEM, SEE PAGE H, BVH IS SOLUTION VECTOR
     CALL SOLV2( A, GVEC,BVH,AD, BD, IFD, IPVT, FLU, RES,
     *           DX,XR,UL,EQUIL,NPOINT,NITER,GME,FN,UDOT,VL1,M,IS,IC,
     *           XLAM,FVEC)
  231  CONTINUE
       RETURN
       END

C *** FPREP, UP FRONT DATA SETUP FOR CALC. UP TO ASEEK MATRIX
```

```
      SUBROUTINE FPREP(G, ALP, ISA, JS, IC, JC, UTER,OPGI, ZI,
     *      Z,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,E11,E12,E13,E32,E33,FL2I,
     *      A, VL2, S, AD, BD, IFD, IPVT, FLU, RES, DX, XR, UL,
     *      EQUIL, C, CI, FK, BM, PG, OPGJ, EALP, FN, VL1,IFD1,
     *      IS, N6,M, N3,NPOINT, NUNU, I9OUT,IFLAG,B,A1,BI,VL21)
      IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION G(1), ALP(1),ISA(1), JS(1), A1(IFD,1),
     *      UTER(1),  Z(1), E11(M,1), E13(M,1), E32(IS,1), E12(M,1),
     *      E33(IS,1), FL2I(1), A(IFD,1), VL2(1), S(1), AD(IFD,1),
     *      BD(1), IPVT(1), FLU(1), RES(1), DX(1), XR(1), BI(1),
     *      UL(1), EQUIL(1), C(N6,1), CI(N6,1), FK(1), BM(1),
     *      PG(N3,1), OPGJ(N6,1), EALP(1), FN(1), VL1(1), B(IFD,1),
     *      VL21(1), OPGI(N6,1), ZI(3,3), ZI1(3,3)
      CALL CALG(G, ALP, IS, N6)
      CALL CAINT(G, ISA, JS, IC, JC, N6,IFD1)
      CALL PRMAT(ISA,IC,1)
      CALL PRMAT(JS,JC,1)
        IF (IC .GT. 0) IFLAG=3
        IF (IC .EQ. 0) IFLAG=0
C *** DEFINE PG
      CALL DPG(PG,N3)
C *** DEFINE U VECTOR WITH CURRENT VALUES, UTER
      CALL DUVEC(UTER,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,ALP,Z)
C *** FIND NEW ROOT
C *** DEFINE A
      CALL DFA(A,ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,
     *  E11, E12, E13, E32, E33, IS, M, N3, IFD)
          WRITE(6,76)
 76        FORMAT(1H ,'-----THE A MATRIX ----')
        CALL PRMAT(A,IS,IS)
C *** DEFINE VECTOR L2 UNDER ITERATION LOOP
      CALL DFL2(ALP,Z,Y1,Y2,Y5,P1,P3,FL2I)
         IF(I9OUT.GT.0) WRITE(6,23)
  23  FORMAT(1H ,'SEE PAGE G, L2 VECTOR, SHOULD NEVER CHANGE')
         CALL PRMAT(VL2,IFD,1)
       IF (IFLAG .EQ. 3) GO TO 30
      DO 1 I=1,IS
   1   BI(I)=0.0


C *** SOLVE THE SYSTEM
      CALL SOLV1(A, VL2, S,  AD, BD, IFD, IPVT, FLU, RES,
     *           DX, XR, UL, EQUIL, NPOINT, NUNU,M,IS,IC,FN,VL1,
     *           XLAM,UDOT,FVEC,GME)

C *** DEFINE MATRIX C, SEE PAGE H, C IS N6 X N6
  30  CALL DFC(C,G,JS,JC,N6)
      IF(I9OUT.GT.0) CALL PRMAT(C,N6,N6)
C *** INVERT MATRIX C
```

```
        CALL CINV(CI, C, N6)
         IF(I9OUT.GT.0) CALL PRMAT(G,N6,1)
C *** DEFINE MATRIX K, SEE PAGE H, K IS N6 X 1
        CALL DFK(FK, G, Z, N6)
           IF(I9OUT.GT.0) CALL PRMAT(FK,N6,1)
C *** MULT. C INVERSE X K TO FORM BM, BM IS N6 X 1
        CALL MMULT(BM, CI, FK, N6, N6, 1)
C *** MULT. BM X 1 MATRIX BY .5
         SCALE = 0.5
        CALL MSCLE(BM,SCALE,N6,1)
           IF(I9OUT.GT.0) CALL PRMAT(BM,N6,1)
C *** DEFINE OPERATOR DELGI
          WRITE(6,40)JC
   40      FORMAT(1H,'INACTIVE JC = ',4X,I5)
         CALL GOPJ(JS , JC, PG, OPGJ, N6, N3)
C *** OPGJ IS (N3,JC), JC MUST = 6
C ****       I9OUT = 0
           IF(I9OUT.GT.0) CALL PRMAT(OPGJ, N3, JC)
           IF (IFLAG .EQ. 0) GO TO 5
            CALL SHFTBM(BM,JC,JS)

C *** DEFINE EALP
   5     CALL MMULT(EALP, OPGJ, BM, N3, JC, 1)
           IF(I9OUT.GT.0) CALL PRMAT(EALP, N3, 1)
           IF (IFLAG .EQ.0) GO TO 35
C ***   SOLVE BI FOR ACTIVE CONSTRAINTS
             SCALE=0.5
             CALL DFBI(BI,ZI,Z,FK,ISA,IC,N3,SCALE)
             CALL PRMAT(ZI,3,3)
             CALL GOPI(ISA,IC,PG,OPGI,N6,N3)
             CALL PRMAT(OPGI,N3,IC)
C ***    SOLVE SYSTEM WITH B MATRIX (ACTIVE CONSTRAINTS)
             CALL DFL2A(VL2,VL21,N6,IFD)
             CALL PRMAT(VL21,IFD,1)
             CALL DFA1(OPGI,A1,IC,M,IS,N6)
             CALL DFB(B,A,A1,IS,IC,M,IFD,IFD1,N6)
             CALL PRMAT(B,IFD1,IFD1)
             CALL SOLV1(B,VL21,S,AD,BD,IFD,IPVT,FLU,RES,
     *                  DX,XR,UL,EQUIL,NPOINT,NUNU,M,IFD1,IC,FN,VL1,
     *                  XLAM,UDOT,FVEC,GME)
  35       CONTINUE
C *** DEFINE N, FN
C        CALL GY(FN,P1,P2,P3,P4,P5,ALP,NUNU)
C          IF(I9OUT.GT.0) CALL PRMAT(FN,M,1)
C *** DEFINE THE L1 VECTOR, VL1, OF LENGTH M
C        CALL DVL1(VL1, Y1, Y2,Y3,Y4,Y5, P1, P2,P3,P4,ALP)
          RETURN
          END
C **** FUDOT, CALC. THE INITIAL UDOT
```

```
      SUBROUTINE FUDOT(UDOT, FPRIM, A, C, OPGJ, FVEC, VL1,ISA,JS,
     *           FN,FL2,FK,ALP,Z,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,FPF,S,
     *            FV00, UU1, ASEEK, AD, WORK, FVNT, FVIT, SIGMA,IFD1,
     *           ISPACE, UDOTD, XLAM, DELTA, SIGN9,IC,JC,OPGI,ZI,
     *           KA1,M, IS,N3, NRF,IFD, N6,I9OUT, NPOINT, NITER,IFLAG)
      IMPLICIT REAL * 8 (A-H,O-Z)
      DIMENSION UDOT(1), FPRIM(NRF,1), A(IFD,1), C(N6,1), IC(1),
     *           OPGJ(IS,1), FVEC(1), VL1(1), FN(1), FL2(1), JC(1),
     *           FK(1), ALP(1), Z(1), FPF(1), S(1), OPGI(N6,1),ISA(1),
     *           FV00(1), UU1(1,1), ASEEK(KA1,1), AD(KA1,1), ZI(3,3),
     *           WORK(1), FVNT(1), FVIT(1), ISPACE(1), UDOTD(1), JS(1)
      CALL DFPRIM(FPRIM, A, C, NRF, IFD, N6, OPGJ,M,IS,IC,JC,OPGI,
     *         ZI,IFLAG,IFD1)
      IF(I9OUT.GT.0) WRITE(6,311)
 311  FORMAT(1H ,'---- THE F PRIME MATRIX ---')
      IF(I9OUT.GT.0) CALL PRMAT(FPRIM, NRF, NRF)
C *** DEFINE THE F VECTOR ON PAGE J WITH FOUR CALLS TO FILLV
C *** F VECTOR, FVEC IS 2M + IS + N6 IN LENGTH
C     CALL FILLV(FVEC, VL1, 0, M)
C     CALL FILLV(FVEC, FN,  M, M)
      CALL FILLV(FVEC, FL2, 0, IS)
      CALL FILLV(FVEC, FK, IS,N6)
      IF(I9OUT.GT.0) WRITE(6,201)
 201  FORMAT(1H ,'----THE F VECTOR ------')
      IF(I9OUT.GT.0) CALL PRVEC(FVEC,NRF,1)
C *** MULT. FPRIM BY F
C ***    CALL MMULT(FPF, FPRIM, FVEC, NRF, NRF, 1)
C *** DANS METHOD, (FPRIM INVERSE)(F) = LINEAR SOLUTION FROM PAGE G
      DO 6 I = 1,NRF
      FPF(I) = 0.0
      IF(I.LE.IFD) FPF(I) = S(I)
  6   CONTINUE
C *** CALC. NORMAL
      CALL MNORM(FPF, XN, NRF, 1)
      IF(I9OUT.GT.0) WRITE(6,202)
 202  FORMAT(1H ,'----THE A VECTOR USED IN DEF. OF UDOT ---')
      IF(I9OUT.GT.0) CALL PRVEC(FPF,NRF,1)
C *** CALC. LAMBDA DOT
      XLAM = 1.0/DSQRT(1.0D0 + XN * XN)
        XLAM = DABS(XLAM)
        IF(SIGN9.LT.0.0) XLAM = -XLAM
C *** CALC. U DOT, (MATRIX) X (SCALAR)
      DO 2 I = 1,NRF
      UDOT(I) = FPF(I)
  2   CONTINUE
C ***    IF(I9OUT.GT.0) CALL PRVEC(UDOT,NRF,1)
      CALL MSCLE(UDOT,XLAM, NRF, 1)
C *** CALC. UTRAN X UZERO + XLAM
      IF(I9OUT.GT.0) WRITE(6,203)
```

```
 203      FORMAT(1H ,'----UDOT ------')
           IF(I9OUT.GT.0) CALL PRVEC(UDOT,NRF,1)
          DO 1 I = 1,NRF
          FVOO(I) = FVEC(I)
          UU1(1,I) = UDOT(I)
  1       CONTINUE
C         CALL MMULT(UU2, UU1, UZERO, 1   , NRF, 1)
C         UU3 = UU2 + XLAM
C         IF(UU3.GT.0.0) GO TO 10
C         XLAM = -XLAM
C         DO 9 I =1,NRF
C         UDOT(I) = -UDOT(I)
C  9      CONTINUE
C 10      CONTINUE
           IF(I9OUT.GT.0) CALL PRVEC(FVEC,NRF,1)
            IF(I9OUT.GT.0) WRITE(6,18) (UU1(1,I),I=1,NRF)
   18       FORMAT(1H ,'UDOT TRANSPOSE'/(1H ,1P,10E12.4))
             IF(I9OUT.GT.0) WRITE(6,103) XLAM
 103       FORMAT(1H ,'*** PARTM CALLING CSIGMA **, LAMBDA DOT',1P,E15.5)
C *** CALC. SIGMA
          CALL CSIGMA(SIGMA, Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,ALP,Z,
     *     FPRIM,ASEEK,FVEC,UDOT,AD,WORK,FVNT,FVIT,A,ISA,JS,
     *     DELTA, N3, IFD, NRF, N6,ISPACE,KA1, UDOTD, M, XLAM,FV00,
     *     I9OUT)
C ***       CALL DSIGMA(SIG2,UZERO,UDOT)  ... GET UZERO
          IF(I9OUT.GT.0) WRITE(6,101) SIGMA, XLAM, SIG2
 101      FORMAT(1H ,'CSIGMA COMPLETE, SIGMA=',1P,E15.5,2X,'LAMBDA DOT'
     *    ,1P,E15.5/1H ,'SIGMA BY 2ND DERV.',1P,E15.5)
          RETURN
          END
C *** DEFINE THE VECTOR L2, SEE PAGE B
         SUBROUTINE DFL2 (ALP, Z, Y1, Y2,Y5, P1,P3, FL2)
        IMPLICIT REAL * 8 (A-H,O-Z)
         DIMENSION ALP(1), Z(1), FL2(1)
C *** SPECIAL FOR PREC. TEST
            DIMENSION FV1(6)
         FL2(1)=2* ALP(1) + 40 +Z(1)+ Z(2)+ Z(3)
         FL2(2)=2* ALP(2) + 20 + Z(2) + Z(3)
         FL2(3)=2* ALP(3) +Z(3)
 2       CONTINUE
         RETURN
         END
C *** AROOT, SETUP TO SEEK ANOTHER ROOT
      SUBROUTINE AROOT(ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,Z,CGRK,SIG3,N6,ISA)
C *** DEF. OF ALPS APPROX. AS OF 4-25-84
       IMPLICIT REAL * 8 (A-H,O-Z)
         COMMON /TRACK1/ NPOINT, NUNU
         DIMENSION ALP(1), Z(1), SIG3(3), ISA(1)
         THETA = (SIG3(1) - SIG3(3)) / (SIG3(2) - SIG3(3))
```

```
          ALP(1) = ALP(1) * THETA + (1.0 - THETA) * ALP(1)
          ALP(2) = ALP(2) * THETA + (1.0 - THETA) * ALP(2)
          ALP(3) = ALP(3) * THETA + (1.0 - THETA) * ALP(3)
           WRITE(16,2) NPOINT,NUNU,THETA,SIG3,ALP,Z
   2       FORMAT(1H ,'ROOT',2I5/1H,1P,10D12.4/1H ,1P,7D12.4)
          DO 1 I = 1,N6
          J=ISA(I)
          Z(J) = Z(J) * THETA + (1.0 - THETA) * Z(J)
  1       CONTINUE
          CGRK = 0.0
          RETURN
          END
C *** FUZIP, CALC. THINGS THAT ARE A FUNCTION OF UZERO
         SUBROUTINE FUZIP(FMMM, VL1, FL2,ALP,S,FN,FK,FVEC,FPF,FPRIM,
     *  IPVT,UDOTD,A,C,   UDOT,UU1,UZERO,UTER,UDIF,AVEC,DUIT,UZIT,
     *   BVH, BVA,  CGRK, M, IS, N6, IFD, NRF, KA1,ASEEK,OPGJ,OPGI,
     *   G, Z, AD, WORK, FVNT, FVIT, XLZERO, IFIRST,NITER,IFD1,JS,
     *   Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,DSIT,XLAM,DELTA, FV00, ISA,
     *   GAP, FL2I, GME, GVEC, BD, RES, DX, XR, UL, FLU,IC,JC,ZI,
     *   EALP, EQUIL,ISPACE,SIGN9,I9OUT,SCANT,SIGMA,NPOINT,IFLAG)
C *** DEFINE F AND F PRIME, SEE PAGE J
C *** DAN WILL PROVIDE MORE INFORMATION
        IMPLICIT REAL * 8 (A-H,O-Z)
       DIMENSION AD(KA1,1), WORK(1)
C *** TEMPORARY DIMENSIONS FOR PARTM
         DIMENSION FVEC(1), FN(1), FK(1), FPF(1), UDOT(1), S(1),
     *  UU1(1,1), UZERO(1), UTER(1), UDIF(1), AVEC(1),UDOTD(1),
     *  BVH(1), BVA(1), DUIT(1), UZIT(1),FPRIM(NRF,1),OPGJ(IS,1),
     *  VL1(1), FL2(1),A(IFD,1),C(N6,1),IPVT(1),ASEEK(KA1,1),JS(1),
     *  ALP(1), G(1), Z(1), FVNT(1), FVIT(1), FV00(1),ISPACE(1),
     *  GAP(1), FL2I(1), GME(1), GVEC(1), BD(1), RES(1),OPGI(IS,1),
     *  DX(1), XR(1), UL(1), FLU(1), EALP(1), EQUIL(1), IC(1), JC(1),
     *  ZI(3,3), ISA(1)
           DIMENSION ALPF(3)
C *** END OF TEMP.
C ***      I9OUT= 0
            N3 = IS
          CALL DFPRIM(FPRIM,A,C,NRF,IFD,N6,OPGJ, M,IS,IC,JC,OPGI,ZI,
     *         IFLAG,IFD1)
          IF(NITER.GT.1) GO TO 12
          IF(I9OUT.GT.0) WRITE(6,311)
 311      FORMAT(1H ,'---- THE F PRIME MATRIX ---')
           IF(I9OUT.GT.0) CALL PRMAT(FPRIM, NRF, NRF)
C *** DEFINE THE F VECTOR ON PAGE J WITH FOUR CALLS TO FILLV
C *** F VECTOR, FVEC IS 2M + IS + N6 IN LENGTH
C         CALL FILLV(FVEC, VL1, 0, M)
C         CALL FILLV(FVEC, FN,  M, M)
          CALL FILLV(FVEC, FL2, 0, IS)
          CALL FILLV(FVEC, FK, IS,N6)
```

```
   12      CONTINUE
           IF(I9OUT.GT.0) WRITE(6,201)
  201      FORMAT(1H ,'----THE F VECTOR ------')
            IF(I9OUT.GT.0) CALL PRVEC(FVEC,NRF,1)
           DO 11 I = 1,NRF
           UU1(1,I) = UDOT(I)
   11      CONTINUE
             IF(I9OUT.GT.0) WRITE(6,18) (UU1(1,I),I=1,NRF)
    18       FORMAT(1H ,'UDOT TRANSPOSE'/(1H ,1P,10E12.4))
             IF(I9OUT.GT.0) WRITE(6,103) XLAM
  103       FORMAT(1H ,'*** PARTM CALLING CSIGMA **, LAMBDA DOT',1P,E15.5)
            IF(SCANT.GT.0.0) GO TO 8
C *** CALC. SIGMA
          CALL CSIGMA(SIGMA, Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,ALP,Z,
     *     FPRIM,ASEEK,FVEC,UDOT,AD,WORK,FVNT,FVIT,A, ISA, JS,
     *     DELTA, N3, IFD, NRF, N6,ISPACE,KA1, UDOTD, M, XLAM,FV00,
     *     I9OUT)
C ***       CALL DSIGMA(SIG2,UZERO,UDOT)  ... GET UZERO
           IF(I9OUT.GT.0) WRITE(6,101) SIGMA, XLAM, SIG2
  101      FORMAT(1H ,'CSIGMA COMPLETE, SIGMA=',1P,E15.5,2X,'LAMBDA DOT'
     *     ,1P,E15.5/1H ,'SIGMA BY 2ND DERV.',1P,E15.5)
    8      CONTINUE
           RETURN
           END
C *** DUTER, CALC INITIAL UTER AND XI
           SUBROUTINE DUTER(UTER, CGRK, XLZERO, SIGMA, XLAM,IS,N6,
     *                      UDOT, UZERO, NRF, I9OUT,ISA)
         IMPLICIT REAL * 8 (A-H,O-Z)
           DIMENSION UTER(1), UDOT(1), UZERO(1),ISA(1)
             CGRK = XLZERO + SIGMA * XLAM
             DO 15 I = 1,IS
             UTER(I) = UZERO(I) + SIGMA * UDOT(I)
   15        CONTINUE
             DO 16 K=1,N6
             J=ISA(K)
             UTER(J+3) = UZERO(J+3) + SIGMA * UDOT(K+3)
   16        CONTINUE

           IF(I9OUT.GT.0) WRITE(6,212)
   212     FORMAT(1H ,' THE UTER VECTOR, V VECTOR ITERATION DEP. ')
             IF(I9OUT.GT.0) CALL PRVEC(UTER,NRF,  1)
            RETURN
            END
C *** UTUPD , UPDATE BASIC DATA FROM UTER
         SUBROUTINE UTUPD(UTER,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,ALP,Z,N3,N6)
         IMPLICIT REAL * 8 (A-H,O-Z)
           DIMENSION  UTER(1), ALP(1), Z(1)
           DO 3 I = 1,N3
           ALP(I) = UTER(I)
```

```
    3       CONTINUE
            DO 2 I = 1,N6
            Z(I) = UTER(I+3)
    2       CONTINUE
            RETURN
            END
C *** FPZIP, UP FRONT DATA SETUP FOR PREP. FOR FUZIP
          SUBROUTINE FPZIP(G, ALP, ISA, JS, IC, JC, UTER,OPGI,
     *         Z,P1,P2,P3,P4,P5,Y1,Y2,Y3,Y4,Y5,E11,E12,E13,E32,E33,FL2I,
     *         A, VL2, S, AD, BD, IFD, IPVT, FLU, RES, DX, XR, UL,
     *         EQUIL, C, CI, FK, BM, PG, OPGJ, EALP, FN, VL1,
     *         IS, N6, M, N3, IFD, NPOINT, NUNU, I9OUT, IFLAG)
         IMPLICIT REAL * 8 (A-H,O-Z)
          DIMENSION G(1), ALP(1), ISA(1), JS(1),
     *         UTER(1),  Z(1), E11(M,1), E13(M,1), E32(IS,1), E12(M,1),
     *         E33(IS,1), FL2I(1), A(IFD,1), VL2(1), S(1), AD(IFD,1),
     *         BD(1), IPVT(1), FLU(1), RES(1), DX(1), XR(1),OPGI(N6,1),
     *         UL(1), EQUIL(1), C(N6,1), CI(N6,1), FK(1), BM(1),
     *         PG(N3,1), OPGJ(N6,1), EALP(1), FN(1), VL1(1)
         CALL CALG(G, ALP, IS, N6)
         CALL CAINT(G, ISA, JS, IC, JC, N6, IFD1)
          IF(IC .GT. 0) IFLAG=3
          IF (IC .EQ. 0) IFLAG=0
C *** DEFINE PG
         CALL DPG(PG,N3)
C *** DEFINE A
         CALL DFA(A,ALP,Y1,Y2,Y3,Y4,Y5,P1,P2,P3,P4,P5,
     *   E11, E12, E13, E32, E33, IS, M, N3, IFD)
             IF(I9OUT.GT.0) WRITE(6,76)
 76            FORMAT(1H ,'-----THE A MATRIX ----')
           IF(I9OUT.GT.0) CALL PRMAT(A,IS,IS)
C *** DEFINE MATRIX C, SEE PAGE H, C IS N6 X N6
         CALL DFC(C,G,JS,JC,N6)
         IF(I9OUT.GT.0) CALL PRMAT(C,N6,N6)
C *** INVERT MATRIX C
         CALL CINV(CI, C, N6)
          IF(I9OUT.GT.0) CALL PRMAT(G,N6,1)
C *** DEFINE MATRIX K, SEE PAGE H, K IS N6 X 1
         CALL DFK(FK, G, Z, N6)
            IF(I9OUT.GT.0) CALL PRMAT(FK,N6,1)
C *** DEFINE OPERATOR DELGI
          CALL GOPJ(JS , JC, PG, OPGJ, N6, N3)
C *** OPGJ IS (N3,JC), JC MUST = 6
C ****       I9OUT = 0
           IF(I9OUT.GT.0) CALL PRMAT(OPGJ, N3, JC)
           CALL GOPI(ISA,IC,PG,OPGI,N6,N3)
           RETURN
           END
```

```
C ***    TO SOLVE SYSTEM FOR Y'S BY L1=0(D1N*Y=-D1R) ***
         SUBROUTINE YSOLV(P1,P2,P3,P4,ALP,M,X)
          IMPLICIT REAL *8 (A-H,O-Z)
          DIMENSION DN1(5,5),OLU(5,5),EQUIL(5),X(5),B(5)
          DIMENSION IPVT(5)

          M=5
          DO 1 I=1,M
           DO 1 J=1,M
            OLU(I,J)= 0.0D0
            DN1(I,J)=0.0D0
  1       CONTINUE
          CALL DON1(P1,P2,P3,P4,DN1,M,ALP)
          IDGT=5
          CALL LUDATF(DN1,OLU,M,M,IDGT,D1,D2,IPVT,EQUIL,WA,IER)
          DO 2 I=1,M
  2         B(I)=0.0D0
          B(3)=-1.0D0
          CALL LUELMF(OLU,B,IPVT,M,M,X)

          RETURN
          END

C *** SOLVE L2 FOR Z'S - ACTIVE CONSTRAINTS ***
          SUBROUTINE ZSOLV(ALP,N6,W)
          IMPLICIT REAL*8 (A-H,O-Z)
          DIMENSION ALP(1), DG(3,3), UGL(3,3), EQUIL(3)
          DIMENSION IPVT(3), W(3), RBG(3)
            CALL FILLZ(DG,N6,N6)
            CALL DPG(DG,N6)
            CALL FILLZ(UGL,N6,N6)
            IDGT =3
            CALL LUDATF(DG,UGL,N6,N6,IDGT,D1,D2,IPVT,EQUIL,WA,IER)
            RBG(1) = -2. * ALP(1) - 40.
            RBG(2) = -2. * ALP(2) - 20.
            RBG(3) = -2. * ALP(3)
            CALL LUELMF(UGL,RBG,IPVT,N6,N6,W)
          RETURN
          END


C *** SOLVE FOR RIGHT HAND SIDE BI FOR ACTIVE CONSTRAINTS
          SUBROUTINE DFBI(BI,ZI,Z,FK,ISA,IC,N3,SCALE)
          IMPLICIT REAL*8 (A-H,O-Z)
          DIMENSION BI(1),ZI(3,3),Z(1),ISA(1),FK(1),ZI1(3,3)
           CALL FILLZ(ZI,3,3)
           DO 1 I=1,IC
           J=ISA(I)
```

```
            ZI(I,I)= Z(J)
            IF (ZI(I,I) .NE.0.0) GO TO 2
            BI(I)=0.0
            GO TO 1
 2          ZI1(I,I) = 1.0D0/ZI(I,I)
            BI(I)=FK(J)*ZI1(I,I)*SCALE
 1          CONTINUE
           RETURN
           END
          SUBROUTINE SHFTBM(BM,JC,JS)
           IMPLICIT REAL *8 (A-H,O-Z)
           DIMENSION BM(1),JS(1)
           DO 5 I=1,JC
             J=JS(I)
             BM(I)=BM(J)
 5         CONTINUE
          RETURN
          END
```

VITA

Gypsy Lynn Tweed was born in Greeneville, Tennessee on March 4, 1963. She attended Glenwood and Mosheim elementary schools in that city before moving with her family to Evansville, Indiana where she attended F. J. Reitz High School for one year. Upon transfer of her family to Bristol, Tennessee, she attended Bristol Tennessee High School during her sophomore and junior years. After relocation of her family to Smyrna, Tennessee, she graduated from Smyrna High School in May 1981. In the fall of 1981, she entered Middle Tennessee State University in Murfreesboro, Tennessee, and in December 1985 she graduated with a Bachelor of Science degree in Physics. The following fall she accepted a research assistantship from the Nuclear Engineering Department at The University of Tennessee, Knoxville and began study toward a Master of Science degree which was awarded in December 1988. During this period, she also performed research sponsored by the Engineering Physics and Mathematics and Instrumentation and Controls Divisions of the Oak Ridge National Laboratory.

The author is a member of the American Nuclear Society and Sigma Pi Sigma. Ms. Tweed will be employed by The Savannah River Plant in Aiken, South Carolina after graduation.