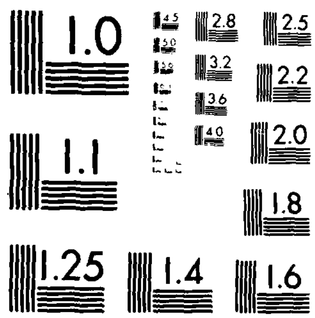
AD-A110 584 PITTSBURGH UNIV PA.INST FOR COMPUTATIONAL MATHEMATICS--ETC F/6 12/1
A PROGRAM FOR A LOCALLY-PARAMETRIZED CONTINUATION PROCESS.(U)
OCT 81 W C RHEINBOLDT, J V BURKARDT N00014-77-C-0623
UNCLASSIFIED ICMA-81-30 NL

1 OF 1
AD A110584

END
DATE
FILMED
8-82
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

LEVEL II

12

INSTITUTE FOR COMPUTATIONAL MATHEMATICS AND APPLICATIONS

Department of Mathematics and Statistics

University of Pittsburgh

Technical Report ICMA-81-30 October 1981

A Program for a Locally-Parametrized Continuation Process*)

by

Werner C. Rheinboldt and John V. Burkardt

Institute for Computational Mathematics and Applications
Department of Mathematics and Statistics
University of Pittsburgh
Pittsburgh, PA 15261
U.S.A.

*) This work was supported in part by the National Science Foundation under Grant MCS-78-05299 and the Office of Naval Research under Contract N00014-77-C-0623.

1. INTRODUCTION

The study of many equilibrium phenomena leads to non-linear equations which involve a number of intrinsic parameters. Interest then centers rarely on the determination of a few specific solutions of the equations for fixed parameter values but rather on an assessment of the behavior of these solutions under general variations of the parameters. For example, in structural analysis the parameters may characterize load points and load directions, material properties, geometrical data, etc. The set of all solutions and associated parameter values has been called the equilibrium surface of the structure (see eg. [31]). This equilibrium surface provides considerable insight into the behavior of the structure and the stability properties (see eg. [23], [32] for further discussions and various examples). From a numerical viewpoint the question then is to analyze computationally the shape and characterize particular features of this equilibrium surface.

In nonlinear mechanics the principal tools for such a computational analysis are the so-called incremental methods. These procedures were developed more or less independently in the engineering literature. But they are now also recognized to be closely related to the continuation methods used for some time in mathematics in general and in numerical analysis in particular. The literature in this area is extensive; we refer only to [21] for a discussion about the connection between incremental approaches for structural problems and continuation methods, to [8] for a historical overview of uses of continuation techniques in mathematics and to [2], [35] for some literature survey of numerical aspects of continuation methods.

Not surprisingly there are differences between the methods used in structural engineering and numerical analysis and neither is directly suited to the analysis of an equilibrium surface. In the numerical analysis literature continuation methods are usually considered only as tools for determining a specific solution y^* of a given nonlinear operator equation $Gy = 0$. For this the equation is imbedded into a one-parameter family $H(y,t) = 0$ which has a solution $y = y(t)$ for each fixed t in some interval, say, $0 \leq t \leq 1$. (See eg. [15] for a survey of such imbeddings.) If $y(t)$ depends continuously on t and satisfies $y(0) = y^0$ and $y(1) = y^*$, where y^0 is a known point, then the numerical process constructs a sequence of points in the proximity of the path $y(t)$, $0 \leq t \leq 1$, starting at y^0 and ending at the desired point y^*. On the other hand, in structural mechanics incremental methods usually are designed to follow numerically a specific load curve parametrized by a load intensity. Hence, while in the imbedding approach the parameter is essentially artificial, in the incremental procedures it has an intrinsic meaning for the application, and, even more importantly, there is no longer a fixed endpoint which is the aim of the computation, but the load curve itself is of interest.

For a numerical analysis of a given equilibrium surface we need to consider continuation-methods in a broader sense as a collection of numerical procedures for completing at least the following three basic tasks:

(i) Follow numerically any curve on the surface specified by a particular combination of parameter values with one degree of freedom.

(1.1) (ii) On any such curve determine the exact location of target points where a given state variable has a specified value.

(iii) On such a curve identify and compute exactly the critical points where stability may be lost.

Beyond this various more special tasks may arise as, for example, the following ones:

(iv) From any one of the critical points determined under (iii) follow a path in the critical boundary.

(1.2) (v) On any one of the curves (i) determine the location of bifurcation points and the paths intersecting at that point.

Methods which are either directly applicable or can be readily adapted to completing these various tasks have been proposed by various authors. In particular, for (i) the literature is very large and we refer here only to the mentioned surveys [2], [35]. Methods relating to (iii) were described, for instance, in [1], [20], [22], [33], [34], and for (iv) and (v) we refer to [28] and [10], [25], respectively, where also further references are given.

So far only a few library programs for performing these various tasks have been published. Without claim for completeness we mention here [14], [38]. Each one of these programs has the objective of computing a specified solution curve of a nonlinear equation by a continuation approach along the lines sketched above. In this paper, we present a new library package specifically written with the objective of completing the three basic tasks (1.1) (i), (ii), (iii). The package can be expanded to incorporate facilities for (1.2) (iv), (v), but this will not be addressed here. The package is based on the continuation approaches introduced in [26], [27] and incorporates some of the concepts of steplength determination discussed in [7]. At the same time, new techniques of parameter adaptation are utilized here based on a prediction of changes in the curvature of the continuation path.

As with all programming packages further improvements are possible. For example, it is planned to introduce an automatic first step selection and a function-scaling option. Special versions incorporating facilities for the tasks (1.2) are also being designed. But since all these changes are built on the present package the presentation of a documentation of PITCON in its basic form appeared desirable and justified.

2. BASIC FORMULATION

Generally, after suitable discretizations, the equilibrium problems mentioned in the introduction lead to a finite-dimensional, non-linear equation of the form

$$G(y,p) = 0 \tag{2.1}$$

where $y \in R^m$ is a vector of state variables, $p \in R^r$ a vector of parameters, and $G: R^m \times R^r \to R^m$ a given function. Then we are interested in the features of the set

$$E(G) = \{(y,p) \in R^m \times R^r;\ \ G(y,p) = 0\} \tag{2.2}$$

of all solutions of (2.1). Under well-known conditions $E(G)$ represents an r-dimensional manifold in $R^m \times R^r$.

In most applications, interest centers on tracing paths on $E(G)$ which are characterized by $r-1$ relations between the parameters. In other words, we are given a suitable mapping $K: R^r \to R^{r-1}$ and wish to compute the subset of $E(G)$ defined by the augmented equations

$$\begin{aligned} G(y,p) &= 0, \\ Kp &= 0. \end{aligned} \tag{2.3}$$

In this formulation we should include the parameters in the list of variables, in which case,(2.3) represents a system with one more variable then equations. Then, for ease of notation, it is reasonable to combine the vectors y and p into one vector x of dimension $n = m+r$. Moreover, from the viewpoint of our package of library programs it is natural to assume that both mappings

G and K of (2.3) are provided for by the user. In other words, we may write (2.3) as one equation

$$(2.4) \qquad Fx = 0$$

with a user-specified mapping $F: R^n \to R^{n-1}$. Note, however, that in this underdetermined equation (2.4) no one variable is explicitly identified as continuation variable as is typical in the incremental and continuation methods mentioned in the introduction.

We assume here that the given mapping F has the following properties:

(2.5)

(i) F is continuously differentiable on R^n.

(ii) The derivative $DF(x)$ of F is locally lipschitzian on R^n.

(iii) The regularity set $R(F) = \{x \in R^n;\ \text{rank } DF(x) = n-1\}$ is non-empty and therefore an open subset of R^n.

From (2.5) it follows (see [26]) that the tangent map specified by

$$(2.6) \qquad T: R(F) \to R^n, \quad DF(x)Tx = 0, \quad ||Tx||_2 = 1, \quad \det\begin{pmatrix} DF(x) \\ (Tx)^T \end{pmatrix} > 0$$

is uniquely determined and locally lipschitzian on $R(F)$. *Furthermore*, (2.5) implies that the regular solution set $E(F) \cap R(F)$ of F is either empty or a one-dimensional C^1-manifold in the open set $R(F)$. Our objective is to determine numerically a non-empty connected component E^* of $E(F) \cap R(F)$. It is well-known (see eg. [17]) that such a component E^* is diffeomorphic either to the circle or to some interval (that is, some connected subset) of R^1. Hence, E^* is uniquely determined by any one of its points $x^0 \in E(F) \cap R(F)$

and we denote this by writing $E^*(F,x^0)$., Note that for any $x^1 \in E^*(F,x^0)$ we have $E^*(F,x^1) = E^*(F,x^0)$.

A parametrization by arclength of $E^*(F,x^0)$ is a solution of the initial value problem

$$(2.7) \qquad \dot{x} = Tx, \quad x(0) = x^0.$$

Note that, since T is locally Lipschitzian, (2.7) has a unique solution which cannot terminate inside $R(F)$. Evidently standard ODE-solvers may be applied to solve (2.7) numerically. This has been pursued for some time in the literature (see eg. [3], [6], [13], [37]). Independent of this, the choice of the arclength for the parametrization of $E^*(F,x^0)$ has been proposed by many authors. Notably H. B. Keller and his co-workers (see eg. [10], [11]) have advocated this choice for some time. It is also the basis of incremental procedures given in [5], [29] and has been more or less implicit in various papers in the field.

Our programs here are based more generally on the structure of $E^*(F,x^0)$ as a one-dimensional manifold and use a local parametrization at each point computed along $E^*(F,x^0)$. A natural class of such local parameters are the n components of the vector x. We call a process based on this choice of parametrization a locally-parametrized continuation method.

3. OUTLINE OF THE PROCESS AND BASIC STEPS

As noted before, our objective is to determine numerically a non-empty component $E^*(F,x^0)$ of the regular solution set $E(F) \cap R(F)$. For the discussion it is useful to consider a parametrization by arclength of $E^*(F,x^0)$, that is, a function $x: J \to E^*(F,x^0)$ which maps some interval $J \subset R^1$ diffeomorphically onto some open subset of $E^*(F,x^0)$ such that $||\dot{x}(s)||_2 = 1$ for $s \in J$. We may assume also that $x(0) = x^0$, $0 \in J$.

The process described here belongs to the class of predictor-corrector continuation methods. Starting from x^0 it produces a sequence of approximations $x^k \doteq x(s_k)$, $k = 0,1,\ldots,$ corresponding to some sequence $0 = s_0 < s_1 < s_2 < \ldots$ of arclength values. Note, however, that in general the values $s_1, s_2, \ldots$ are only approximately computable and are of limited interest in most applications.

In our program the principal steps performed during one continuation step are as follows:

1. Initialization.
2. Check for and computation of target point, if desired.
3. Calculation of tangent vector and determination of new local continuation parameter.

(3.1)

4. Check for and computation of limit point, if desired.
5. Steplength computation.
6. Computation of predicted point and corrector iteration.
7. Storage of data and return.

The sequencing of these steps is dictated by the data-flow. For the description of the details it will be advantageous not to adhere to this sequence. Instead, in the remainder of this section, we discuss the basic steps 3. and 6. Then the next section introduces the new steplength computation used in step 5. and section 5 covers steps 2. and 4. The data-handling steps 1. and 7. should be self-explanatory from the documentation of the program itself.

Let $e^1,\ldots,e^n$ be the natural basis vectors of R^n. Then it is readily verified that (see eg. [26])

$$(3.2)\qquad \det\begin{pmatrix} DF(x) \\ (e^i)^T \end{pmatrix} = [(e^i)^T Tx] \det\begin{pmatrix} DF(x) \\ (Tx)^T \end{pmatrix}, \quad \forall\, x \in R(F), \quad i = 1,\ldots,n,$$

where the matrix occurring on the right is non-singular. Hence, for any index i, $1 \le i \le n$, such that $(e^i)^T Tx \neq 0$, the solution $v \in R^n$ of the linear system

$$(3.3)\qquad \begin{pmatrix} DF(x) \\ (e^i)^T \end{pmatrix} v = e^n$$

is uniquely defined. Evidently, then

$$(3.4)\qquad Tx = \sigma \frac{v}{\|v\|_2},$$

and, in line with (2.6), we should set

$$(3.5)\qquad \sigma = \operatorname{sign}(v^T e^i)\,\operatorname{sign}\det\begin{pmatrix} DF(x) \\ (e^i)^T \end{pmatrix}.$$

As long as the solution path remains completely in $R(F)$ this is satisfactory.

But frequently in applications we may encounter a bifurcation point $x^* \notin R(F)$ where several solution paths terminate. For example, using arclength representations we may find solutions $x^j: J_j \subset R^1 \to R(F)$, $j = 1,\ldots,4$, for which $x^j(s)$ tends to x^* when s tends to one of the endpoints of J_j. Moreover, it often happens that there are pairs of these solutions, say, x^1 and x^2 for which $\lim \dot{x}^1(s) = -\lim \dot{x}^2(s)$ at x^*, (see Fig. 1). In other words, if we disregard the direction of the solutions, they appear to form one smooth curve through x^*. In such a case, when the process moves along x^1 toward x^* it usually "jumps" over x^* onto x^2. Then, unless we reverse the sign of σ in (3.4) the tangent will again point toward x^* and the process reverses direction.

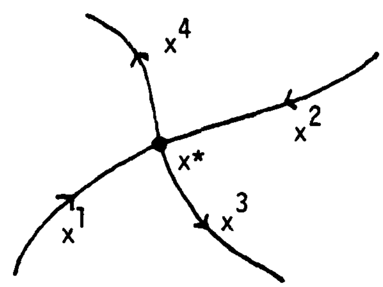


Figure 1

In order to avoid this problem suppose that the point x in (3.4) is the k-th approximation computed along the curve. Then σ is determined as follows

$$\sigma = \begin{cases} \text{dir}, & \text{if } k = 0 \\ +1, & \text{if } \operatorname{sign} v^T e^i = \operatorname{sign}(Tx^{k-1})^T e^i \\ -1, & \text{otherwise} \end{cases} \tag{3.6}$$

where dir is a user specified direction at the starting point. By comparing this value of σ with that of (3.5) we can detect if the process did jump over a bifurcation point of odd multiplicity. Obviously, bifurcation points of even multiplicity cannot be found this way.

Once the tangent Tx^k has been obtained we determine the indices j_1 and j_2 of the largest and second largest component of Tx^k in modulus, respectively. The relation (3.2) certainly suggests that the index i_k, $1 \leq i_k \leq n$, of the new local continuation variable be set equal to j_1. However, if we are approaching a limit point in the j_1-th variable then this choice may be disadvantageous. Accordingly, if the following three conditions are simultaneously satisfied

$$\text{(3.7)} \qquad \begin{aligned} &\text{(i)} \quad |(e^{j_1})^T Tx^k| < |(e^{j_1})^T Tx^{k-1}|, \\ &\text{(ii)} \quad |(e^{j_2})^T Tx^k| > |(e^{j_2})^T Tx^{k-1}|, \\ &\text{(iii)} \quad |(e^{j_2})^T Tx^k| \geq \mu |(e^{j_1})^T Tx^k|, \end{aligned}$$

with a fixed μ, $0 < \mu < 1$, then we set $i_k = j_2$. Of course, if we don't have a previous tangent vector this check has to be bypassed. The new continuation index i_k will be used for the computation of the next point x^{k+1} and its tangent Tx^{k+1}. For the tangent computation at x^0 a continuation index is assumed to be given by the user.

With the tangent Tx^k and the steplength $h_k > 0$ determined by the steplength algorithm of section 4 we compute now the predicted point $\hat{x}^k = x^k + h_k Tx^k$. Then any appropriate iterative method for the solution of the augmented equation

$$\text{(3.8)} \qquad \hat{F}x \equiv \begin{pmatrix} Fx \\ (e^{i_k})^T(x-\hat{x}^k) \end{pmatrix} = 0$$

starting from $\hat{x}^k$ may be used as a corrector process. In the program we use either the regular Newton method or its modified form in which the Jacobian at the starting point is held fixed.

Let $y^0 = \hat{x}^k, y^1, y^2, \ldots$ be the iterates produced in this way. The process has to incorporate provisions for monitoring the convergence and for aborting the iteration as soon as divergence is suspected. In the program non-convergence is declared if any one of the following three conditions is true

$$
\text{(3.9)} \quad
\begin{aligned}
&\text{(i)} && ||\hat{F}y^j|| \geq \theta ||\hat{F}y^{j-1}|| && \text{for some } j \geq 1, \\
&\text{(ii)} && ||y^j - y^{j-1}|| \geq \theta ||y^{j-1} - y^{j-2}|| && \text{for some } j \geq 2, \\
&\text{(iii)} && j \geq j_{max}.
\end{aligned}
$$

For the constant θ we use $\theta = 1.05$ except in the first check of (3.9) (i) where $\theta = 2$ is chosen. The maximal iteration count j_{max} depends on the method. For the regular Newton process we set $j_{max} = 10$ and double this for the modified method. In the case of non-convergence the predictor step is reduced by a given factor, for example 1/3, unless the resulting step is below a given minimal steplength.

Convergence is declared if either one of the two conditions holds for an iterate:

$$
\text{(3.10)} \quad
\begin{aligned}
&\text{(i)} && ||\hat{F}y^j|| \leq 8\, \varepsilon_{mach} \quad \text{for some } j \geq 0, \\
&\text{(ii)} && (||Fy^j|| \leq \varepsilon_{abs}) \text{ and } (||y^j - y^{j-1}|| \leq \varepsilon_{abs} + \varepsilon_{rel} ||y^j||) \quad \text{for some } j \geq 1.
\end{aligned}
$$

The tolerances ε_{abs}, ε_{rel} are user specified and ε_{mach} is the smallest floating point number such that $1. = 1. + \varepsilon_{mach}$. In both tests (3.9) and (3.10) the maximum norm is used.

4. THE STEPLENGTH ALGORITHM

For the points x^k, $k = 0,1,\ldots,$ approximating the continuation curve $x\colon J \to E^*(F,x^o)$ the achievable error $||x^k - x(s_k)||$ is solely determined by the termination criterion (3.10) of the corrector process. In contrast to this the standard ODE-solvers involve a corrector equation obtained by extrapolation for which the solutions are not, in general, on the exact curve. As a consequence the available error for the ODE-solvers depends on the history of the process up to that point, and this in turn has a strong influence on the step-selection. On the other hand, for our continuation process any step $h_k > 0$ along the Euler line is acceptable in principle if only the corrector converges from the predicted point $\hat{x}^k$. Moreover, in [26] it was shown that any compact segment of the continuation curve in $R(F)$ has an ε-neighborhood for some $\varepsilon > 0$ in which Newton's method will converge to the curve.

This suggests that we estimate the radius of convergence of the corrector process at the computed points and extrapolate these radii to the next point about to be determined. In practice the estimate of a convergence radius at some continuation point would have to be based on the corrector iterates which led to that point. Unfortunately, as was proved in [7], this represents insufficient information for obtaining such an estimate. On the other hand, an approach was presented in [7] which allows for an assessment of the convergence quality of the particular sequence of corrector iterates.

For details of this approach we refer to the cited article. In brief, let $\{y^i\}$ be a given sequence with limit y^* generated by an iterative process and denote the errors by $e_i = ||y^i - y^*||$, $i = 0,1,\ldots$. The definition of any convergence measure is based on a hypothetical model of the behavior of the errors. For example, if $\{y^i\}$ converges linearly it is reasonable to assume that

(4.1) $$0 \leq e_{i+1} \leq \lambda e_i, \quad i = 0,1,\ldots$$

with some constant λ, $0 < \lambda < 1$, depending on $\{y^i\}$. Suppose now that the process was terminated with the iterate y^{i*}. Then

(4.2) $$\tilde{\lambda} = \tilde{\omega}^{1/(i*-1)}, \quad \tilde{\omega} = \frac{||y^{i*}-y^{i*-1}||}{||y^{i*}-y^{o}||}, \quad i* \geq 2,$$

represents a computable estimate of λ.

In the setting of our continuation process suppose now that the y^i, $i = 0,1,\ldots,$ are the corrector iterates leading from the current predicted point $\hat{x}^k = y^o$ to the new continuation point $x^{k+1} = y^{i*}$. Then

(4.3) $$\delta_k = ||\hat{x}^k - x^{k+1}|| = ||y^o - y^{i*}||$$

is the correction-distance. For the modified Newton method the convergence is indeed linear, and a reasonable aim in the construction of the steps along the curve is to ensure that the number of corrector iterates remains about constant. In other words, we aim at taking always, say, $m*$ corrector steps. Hence, under the heuristic assumption that the error model (4.1) remains valid for some interval of starting errors e_o around δ_k, we should have begun with an "ideal starting error" $\delta_k^* = \theta_k \cdot \delta_k$ such that

(4.4) $$\tilde{\lambda}^{m*} \delta_k^* \doteq \tilde{\lambda}^{i*} \delta_k$$

and therefore

$$(4.5) \qquad \theta_k = \tilde{\lambda}^{i^*-m^*} \equiv \tilde{\omega}^{(i^*-m^*)/(i^*-1)}.$$

In our program we use $m^* = 10$ for the modified Newton method and enforce always that $0.125 \leq \theta_k \leq 8$.

This technique is also readily applicable for Newton's method. In [7] two different hypothetical error models for the Newton process were discussed. Here we use only one of these models, namely the one arising in the attraction theorem formulated in [24]. In essence, under certain conditions about the equation and the desired limit y^* of the Newton process there exists a radius $r^* > 0$ such that for any starting point y^0 in the ball $B(y^*,r^*)$ the relative errors $\varepsilon_i = e_i/r^*$, $i = 0,1,\ldots,$ satisfy

$$(4.6) \qquad 0 \leq \varepsilon_{i+1} \leq \phi(\varepsilon_i), \quad i = 0,1,\ldots, \quad \phi(t) = \frac{t^2}{3-2t}, \quad 0 \leq t \leq 1.$$

The radius r^* depends on global information about the equation and is not accessible. If $0 < \varepsilon_0 < 1$ and the $\{\varepsilon_i\}$ satisfy (4.6) then we have

$$(4.7) \qquad \varepsilon_i \leq \eta_i \equiv \phi^i(\eta_0) \equiv \frac{3}{1 + 2\cosh 2^i\alpha}, \quad i = 0,1,\ldots, \quad \eta_0 = \varepsilon_0,$$

where α is the unique positive solution of

$$(4.8) \qquad \psi(\alpha) = \eta_0, \quad \psi(\alpha) = \frac{3}{1 + 2\cosh\alpha}.$$

Moreover, for any ω, $0 < \omega < 1$, and $i^* \geq 2$ the equation

(4.9)
$$\frac{1}{\psi(\alpha)}\,\phi^{i*-1}(\psi(\alpha)) \equiv \frac{1 + 2\cosh\alpha}{1 + 2\cosh 2^{i*-1}\alpha} = \omega$$

has a unique solution $\alpha > 0$.

Now suppose that $\{y^i\}$ denotes the sequence of Newton iterates and that the process was terminated at y^{i*}. As in the linear case we use the approximation

(4.10)
$$\tilde{\omega} = \frac{||y^{i*}-y^{i*-1}||}{||y^{i*}-y^{o}||} \doteq \frac{e_{i*-1}}{e_o} = \frac{\varepsilon_{i*-1}}{\varepsilon_o} \leq \frac{\eta_{i*-1}}{\eta_o}$$

and compute with this $\tilde{\omega}$ the solution $\tilde{\alpha}$ of (4.9) which gives the estimate $\tilde{\eta}_o = \psi(\tilde{\alpha})$ of ε_o. Now we proceed as before and obtain the factor

(4.11)
$$\theta_k^* = \frac{\delta_k^*}{\delta_k} = \frac{\tilde{\eta}_o'}{\tilde{\eta}_o}$$

for the ideal starting error by determining the unique solution $\tilde{\eta}_o'$, $0 < \tilde{\eta}_o' < 1$, of

(4.12)
$$\phi^{m*}(\tilde{\eta}_o') = \phi^{i*}(\tilde{\eta}_o).$$

Since the iterates ϕ^i are explicitly known the various equations are not difficult to solve numerically. However, for the computation it is more advantageous to introduce a least squares fit of θ_k as a function of $\tilde{\omega}$ for all relevant values of $i*$. In the program we use $m* = 4$ and the approximations for θ_k given in Table 1. Note that as before we restrict θ_k to the interval $0.125 \leq \theta_k \leq 8$.

i^*	$\tilde{\omega} \in [a,b]$		θ_k
	a	b	
2	0.8735115	1	1
	0.1531947	0.8735115	$0.9043128 - 0.7075675 \ln \omega$
	0.03191815	0.1531947	$-4.667383 - 3.677482 \ln \omega$
	0	0.03191815	8
3	0.4677788	1	1
	0.6970123(-3)	0.4677788	$0.8516099 - 0.1953119 \ln \omega$
	0.1980863(-5)	0.6970123(-3)	$-4.830636 - 0.9770528 \ln \omega$
	0	0.1980863(-5)	8
4	0	1	1
5	0.3339946(-10)	1	$1.040061 + 0.03793395 \ln \omega$
	0	0.3339946(-10)	0.125
6	0.1122789(-8)	1	$1.042177 + 0.04450706 \ln \omega$
	0	0.1122789(-8)	0.125
≥ 7	0	1	0.125

Table 1

We turn now to the algorithm for the determination of the steplength $h_k > 0$ along the Euler line $\pi(t) = x^k + t\,Tx^k$ used for the prediction. In order to estimate the distance between $\pi(t)$ and the exact curve $x = x(s)$ we introduce the quadratic Hermite-Birkhoff interpolation polynomial

$$(4.13)\qquad q(t) = x^k + t\,Tx^k + \frac{1}{2} t^2 w^k, \; w^k = \frac{1}{\Delta s_k}(Tx^k - Tx^{k-1}), \quad \Delta s_k = ||x^k - x^{k-1}||_2$$

for which

$$(4.14)\qquad q(0) = x^k, \;\; q'(0) = Tx^k, \;\; q'(-\Delta s_k) = Tx^{k-1}.$$

Since

$$(4.15)\qquad w^k \doteq \int_0^1 x''(s_k - \sigma\Delta s_k)d\sigma = x''(s_k - \tilde{\sigma}\Delta s_k), \quad 0 < \tilde{\sigma} < 1,$$

the quantity

$$(4.16)\qquad ||w^k||_2 = \frac{2}{\Delta s_k}|\sin \frac{1}{2}\alpha_k|, \quad \alpha_k = \arccos\,((Tx^k)^T Tx^{k-1})$$

represents an approximation of the curvature of the exact point at some point between $x(s_{k-1})$ and $x(s_k)$.

It is tempting to derive from q a prediction of the curvature to be expected during the next continuation step. However, a closer computation shows that the value of the curvature of q assumes its maximum $||w^k||_2/\cos^2 \frac{1}{2}\alpha_k$ at $t = -\frac{1}{2}\Delta s_k$ and that for increasing t this value decreases rapidly. For example, at $t = 0$ the curvature of q equals only $||w^k||_2\,|\cos \frac{1}{2}\alpha_k|$ and for positive t no reasonable predictive information can be gained this way.

The relation (4.15) suggests the use of the simple linear extrapolation

$$(4.17a)\qquad \gamma_k^{tent} = ||w^k||_2 + \frac{\Delta s_k}{\Delta s_k + \Delta s_{k-1}}(||w^k||_2 - ||w^{k-1}||_2)$$

for a prediction of the curvature during the next continuation step. However, this value may become negative and accordingly we use instead

$$(4.17b) \qquad \gamma_k = \max\,(\gamma_{min}, \gamma_k^{tent})$$

with a given small $\gamma_{min} > 0$.

Most of the data discussed so far are sketched in Figure 2. In order to derive a formula for the desired predictor step h_k we note that

$$(4.18) \qquad ||q(t) - \pi(t)||_2 = \frac{1}{2}\, t^2 ||w^k||_2$$

represents an estimate of the distance between the Euler line and the exact curve. In fact, for smooth curves the error of this estimate is asymptotically or order three in $\max(|t|, \Delta s_k)$ as this quantity tends to zero. Hence, if we want this distance to be at most equal to a tolerance $\varepsilon > 0$ then we should choose the next step as

$$(4.19) \qquad t = \sqrt{\frac{2\varepsilon}{||w^k||_2}}\,.$$

It is natural to replace the curvature $||w^k||_2$ by the predicted value γ_k of (4.17) and to relate the tolerance ε to the "ideal starting error" δ_k^* obtained earlier. As Figure 2 indicates it is unreasonable to expect $\varepsilon > \Delta s_k$. Hence, we use instead

$$(4.20) \qquad \varepsilon_k = \begin{cases} \varepsilon_{min}\, \Delta s_k & \text{if} \quad \delta_k^* \le \varepsilon_{min}\, \Delta s_k \\ \Delta s_k & \text{if} \quad \delta_k^* \ge \Delta s_k \\ \delta_k^* & \text{otherwise} \end{cases}$$

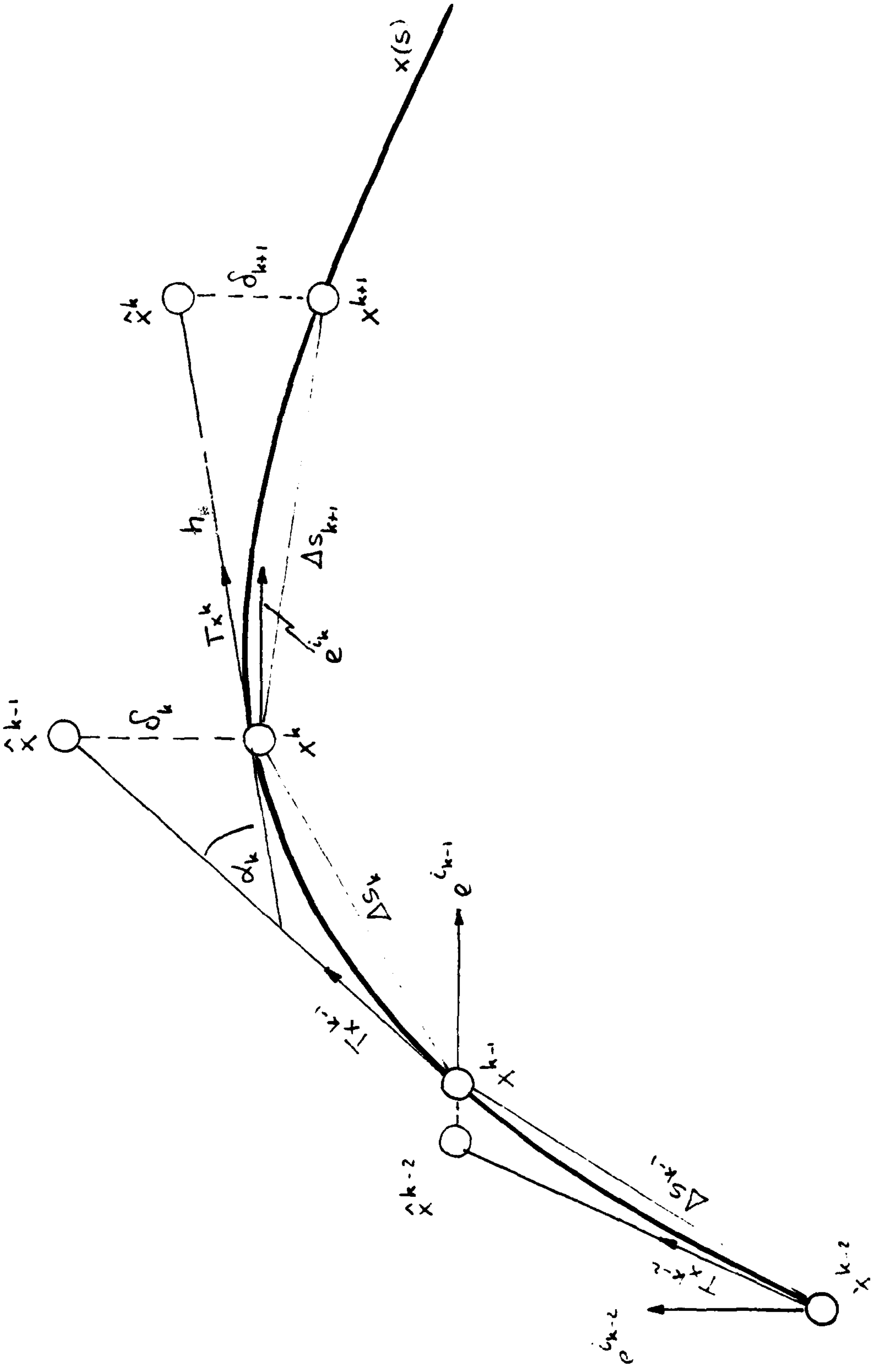

x(s)
x^{k+1}
$\hat{x}^{k}$
δ_{k+1}
h_k
Δs_{k+1}
T_x^{k}
e^{i_k}
$\hat{x}^{k-1}$
δ_k
x^{k}
α_k
Δs_k
$e^{i_{k-1}}$
T_x^{k-1}
x^{k-1}
$\hat{x}^{k-2}$
Δs_{k-1}
T_x^{k-2}
x^{k-2}
$e^{i_{k-2}}$

Figure 2

with a small $\varepsilon_{min} > 0$, e.g., $\varepsilon_{min} = 0.01$. Then a tentative predicted step is given by

$$(4.21) \qquad h_k^{(1)} = \sqrt{\frac{2\varepsilon_k}{\gamma_k}} .$$

From the form (3.8) of the augmented equation we see that the corrector iterates remain in a hyperplane perpendicular to the basis vector e^{i_k} through the predicted point. Then Figure 2 suggests that we adjust the predicted steplength h_k so as to ensure that h_k will be approximately equal to Δs_{k+1}. There is no need to enforce this too rigidly. It suffices to define a new tentative step by the requirement

$$(e^i)^T \pi(h_k^{(2)}) = (e^i)^T q(h_k^{(1)})$$

whence,

$$(4.22) \qquad h_k^{(2)} = h_k^{(1)}[1 + \frac{h_k^{(1)}}{2\Delta s_k} (1 - \frac{(e^i)^T Tx^{k-1}}{(e^i)^T Tx^k})].$$

This formula may involve subtractice cancellation and has to be evaluated in double precision.

The final value h_k of the steplength is now obtained from $h_k^{(2)}$ by enforcing three different bounding requirements. First of all, if the previous continuation step from x^{k-1} to x^k was obtained only after a failure of the correction process and a corresponding reduction of the predicted step, then we should not allow h_k to exceed Δs_k. Secondly, as in the ODE solvers we need to control both the relative growth and the absolute size of the predictor step. Thus, we require that

$$\frac{1}{\kappa}\Delta s_k \leq h_k \leq \kappa\Delta s_k, \quad h_{min} \leq h_k \leq h_{max} \tag{4.23}$$

where κ is some factor, say, $\kappa = 3$, and h_{min}, h_{max} depend on the machine as well as the requirements of the problem. It should be obvious how the final step h_k is obtained from $h_k^{(2)}$ on the basis of these restrictions.

5. THE COMPUTATION OF TARGET AND LIMIT POINTS

By generating a sequence of solution points on a given curve, the continuation process reveals the shape of the curve, but there are often other items of interest that need to be studied as well. Our program is designed to pause during the continuation steps in order to seek out special points that the user has requested, namely, target and limit points.

A target point $x \in E^*(F,x^0)$ is a point on the solution curve for which the component $x_i = (e^i)^T x$ with given index $i = IT$ has a prescribed value $\bar{x}_i = XIT$. Limit or turning points with respect to a given index $i = LIM$ are points $x \in E^*(F,x^0)$ where the i-th component $(e^i)^T Tx$ is zero. More specifically, since it is computationally unreasonable to attempt to compute zeroes of even order, we are concerned only with limit points on the continuation curve where $(e^i)^T Tx(s)$ changes sign.

It might be mentioned that bifurcation points represent another interesting, special class of points. But in that case we are not only interested in the specific location of the point but also in the solution curves that branch off from it. This is exactly the task (1.2) (v) listed earlier. The corresponding procedures (loc. cit.) would add considerably to the complexity of our program, and, since their utility tends to be of a more specialized nature, it was decided not to cover task (1.2) (v) (nor (1.2) (iv)) in the present program.

As indicated before, the determination of a target or limit point represents an interruption in the normal flow of the continuation program. After at least one step has been taken, the program has available an old point x^{k-1}, a new point x^k and the tangent vector Tx^{k-1}. Normally, then we turn to the computation of Tx^k, of the new steplength, and finally of the next point x^{k+1}. But if the index IT or LIM is non-zero then these computations are postponed for the search of a target or limit point, respectively. We discuss these cases separately:

Target points: Suppose that a non-zero value of $i = IT$ and associated value $\bar{x}_i = XIT$ have been given. If $\bar{x}_i$ lies between $(e^i)^T x^{k-1}$ and $(e^i)^T x^k$, then it is assumed that a solution point $x \in E^*(F,x^0)$ with $(e^i)^T x = \bar{x}_i$ is nearby. In this case a point

$$(5.1) \qquad y(t) = (1-t)x^{k-1} + tx^k, \quad 0 \leq t \leq 1,$$

on the secant between x^{k-1} and x^k is determined such that $(e^i)^T y(t) = \bar{x}_i$. Now with the augmenting equation $(e^i)^T x = \bar{x}_i$ the corrector process is applied, and, if it terminates successfully the resulting point is taken as the desired target. Otherwise, a failure is indicated for the target routine. In either case, the routine returns and on the next call the continuation loop will pick up from where it was interrupted. Note that in effect the target routine uses the IT-th variable for the local parametrization of the curve between x^{k-1} and x^k. This may be an inferior choice of parameter for the corrector but it allows us to enforce that the resulting target point $x \in E^*(F,x^0)$ indeed satisfies $(e^i)^T x = \bar{x}_i$. Clearly, for very large continuation steps we have no guarantee that all target points will be detected or that a target computation will succeed. Thus, the utility of the target routine will depend on the maximal allowed stepsize that has been chosen.

Limit points: If the limit point index $i = LIM$ is non-zero, then a limit point determination is carried out after a target point search has been successfully or unsuccessfully completed, provided it was called for at all.

Recall that we still have as current information the vectors x^{k-1}, x^k, and Tx^{k-1}. Now the new tangent Tx^k is evaluated and if $\mathrm{sign}(e^i)^T Tx^{k-1} \neq \mathrm{sign}(e^i)^T Tx^k$ for $i = \mathrm{LIM}\ (\neq 0)$ then a limit point search is begun. For this the index $\hat{i}$ of the largest component in modulus of the secant direction $x^k - x^{k-1}$ is chosen as a local parametrization of the curve between x^{k-1} and x^k. More specifically, suppose that $x\colon [s_{k-1},s_k] \to E^*(F,x^0)$ represents the segment of the curve between x^{k-1} and x^k. Then $\hat{i}$ is assumed to be the index of a local coordinate for which there exists a bijective parameter transformation $\phi\colon [0,1] \to [s_{k-1},s_k]$ such that $(e^{\hat{i}})^T y(t) = (e^{\hat{i}})^T x(\phi(t))$, $0 \le t \le 1$, where $y(t)$ is defined by (5.1).

Hence, we may consider the function

$$(5.2) \qquad g\colon [0,1] \to R^1, \quad g(t) = (e^i)^T Tx(\phi(t)), \quad 0 \le t \le 1,$$

and our problem is to determine a zero of g. Since by assumption $\mathrm{sign}\ g(0) \neq \mathrm{sign}\ g(1)$, a rootfinder of the Dekker-Brent type can be applied. For the evaluation of $g(t)$ we use the augmenting equation $(e^{\hat{i}})^T x = (e^{\hat{i}})^T y(t)$ and apply the corrector process with $y(t)$ as starting point. If it terminates successfully with some x then Tx can be evaluated and we set $g(t) = (e^i)^T Tx$. Hence, g is certainly costly to compute and we require an efficient rootfinder to speed the convergence of the limit point routine. A specially modified version of the routine given in [4] is used in our program. Clearly, as in the case of target points, we may fail to detect a limit point if the continuation steps are too large and in such a situation the rootfinder may also fail to converge. In addition, the evaluation of g may run into difficulties when the desired limit point is near a bifurcation point.

6. SOME NUMERICAL EXAMPLES

The programs described here have been used extensively with excellent success on problems from many different areas. We include here only a few numerical examples to illustrate the operation of the programs.

Example 1. In order to present some details of the performance of the programs we consider first a very small problem which was originally formulated in [9] and subsequently used as a test case by many authors. The mapping F has here the form

$$\text{(6.1)} \qquad Fx = \begin{pmatrix} x_1 - x_2^3 + 5x_2^2 - 2x_2 + 34x_3 - 47 \\ \\ x_1 + x_2^3 + x_2^2 - 14x_2 + 10x_3 - 39 \end{pmatrix}, \quad \forall\, x \in R^3.$$

For the starting point $x^0 = (15,-2,0)^T$ the solution curve passes through $x^* = (5,4,1)^T$ and this point is chosen as target.

Tables 2 and 3 show runs with the full Newton method and modified Newton method, respectively as corrector process. A starting step $h_o = 0.3$ and maximum step $h_{max} = 25.0$ were used. The performance for the two correctors is practically the same although the step-prediction exhibits certain differences due to our assessment of the corrector distance. Clearly, the use of the modified Newton process is much less expensive and hence preferable as Table 4 shows which summarizes the total number of function and Jacobian calls including those for the target calculation. Comparative performance data given in [7] for this problem involved 22 continuation steps, 15 step reductions and 128 Jacobian evaluations. The procedure discussed in [19] required 25 continuation steps but no further details were provided in the paper.

Step	Continuation point x_1	x_2	x_3	Contin. Variable	Total Correct. Steps	Comments
0	15.000	-2.00000	0.00000	x_3	–	
1	14.705	-1.9421	0.065381	x_1	2	
2	14.285	-1.7291	0.26874	x_3	3	
3	16.906	-1.2094	0.54684	x_2	2	
4	24.918	-0.59908	0.55514	x_1	3	
5	48.974	0.71803	-0.080758	x_1	3	
6	57.928	1.2846	-0.40736	x_1	4	Step red.
7	60.052	1.5709	-0.54035	x_1	4	Step red.
8	61.666	2.0010	-0.66683	x_2	2	
9	-5.1039	4.1510	1.3464	x_2	2	Target passed
				TOTAL	25	
				Computation of target	4	

Table 2

Step	Continuation point x_1	x_2	x_3	Contin. Variable	Total Correct Steps	Comments
0	15.000	-2.0000	0.00000	x_3	–	
1	14.710	-1.9421	0.065381	x_1	3	
2	14.285	-1.7291	0.26874	x_3	4	
3	16.906	-1.2094	0.54685	x_2	1	
4	24.918	-0.59906	0.55514	x_1	6	
5	48.975	0.71810	-0.080804	x_1	5	
6	57.289	1.2847	-0.40742	x_1	6	Step red.
7	60.053	1.5711	-0.54042	x_1	5	Step red.
8	61.666	2.0013	-0.66689	x_2	2	
9	-4.4239	4.1413	1.3229	x_2	1	Target passed
				TOTAL	33	
				Computation of target	8	

Table 3

	Corrector Process	
	Newton	Mod. Newton
Function calls	41	53
Jacobian calls	38	21

Table 4

It may be noted that the solution curve has two limit points each with respect to x_1 and x_3. The two step reductions are almost unavoidable here since the curve has a long straight segment followed by a very sharp bend. The target computation is relatively expensive since the last step is extremely large due to another straight curve segment.

Example 2. Maneuvering airplanes, especially at high angles of attack, sometimes undergo sudden jumps in their response to the pilot's control inputs. The problem has been discussed extensively in the literature, see, for example, [18], [30], [39]. Without going into further details we use here a simplified version of a system of five equilibrium equations involving the roll rate (x_1), pitch rate (x_2), yaw rate (x_3), (incremental) angle of attach (x_4), side slip angle (x_5), elevator angle (x_6), aileron angle (x_7), and rudder angle (x_8). More specifically, for the particular aircraft discussed in [18] these equations have the dimensionless form

$$(6.1) \qquad Fx \equiv Ax + \phi(x) = 0, \quad \forall\, x \in R^8$$

where

$$A = \begin{pmatrix} -3.933 & 0.107 & 0.126 & 0 & -9.99 & 0 & -45.83 & -7.64 \\ 0 & -0.987 & 0 & -22.95 & 0 & -28.37 & 0 & 0 \\ 0.002 & 0 & -0.235 & 0 & 5.67 & 0 & -0.921 & -6.51 \\ 1.0 & 0 & 0 & -1.0 & 0 & -0.168 & 0 & 0 \\ 0 & 0 & -1.0 & 0 & -0.196 & 0 & -0.0071 & 0 \end{pmatrix}$$

and

$$\phi(x) = \begin{pmatrix} -0.727\, x_2x_3 + 8.39\, x_3x_4 - 684.4\, x_4x_5 + 63.5\, x_4x_7 \\ 0.949\, x_1x_3 + 0.173\, x_1x_5 \\ -0.716\, x_1x_2 - 1.578\, x_1x_4 + 1.132\, x_4x_7 \\ -x_1x_2 \\ x_1x_4 \end{pmatrix} .$$

Figure 3 shows some solution curves on the three-dimensional equilibrium surface in R^8. More specifically, in all cases we fixed a value of x_6 (elevator deflection) and chose the rudder deflection $x_8 = 0$. The paths $x_6 > \omega_1$, $x_8 = 0$ with $\omega_1 \approx -0.0061771$ have two limit points, for $\omega_1 > x_6 > \omega_2$, $x_8 = 0$, with $\omega_2 \approx -0.012498$ a third limit point appears, and for $\omega_2 > x_6$, $x_8 = 0$ only one limit point remains. A similar picture arises for negative roll rates.

In all cases the programs easily detected and computed the various limit points (see Table 5). But the example also shows that even with a large number of search paths it is difficult to provide a full picture of the location of the critical boundary, that is, of the curves of limit points with respect to x_1 for $x_8 = 0$ and varying x_6, x_7. In Figure 3 the corresponding branches of limit point curves are shown as dotted lines. They were obtained with a code for the

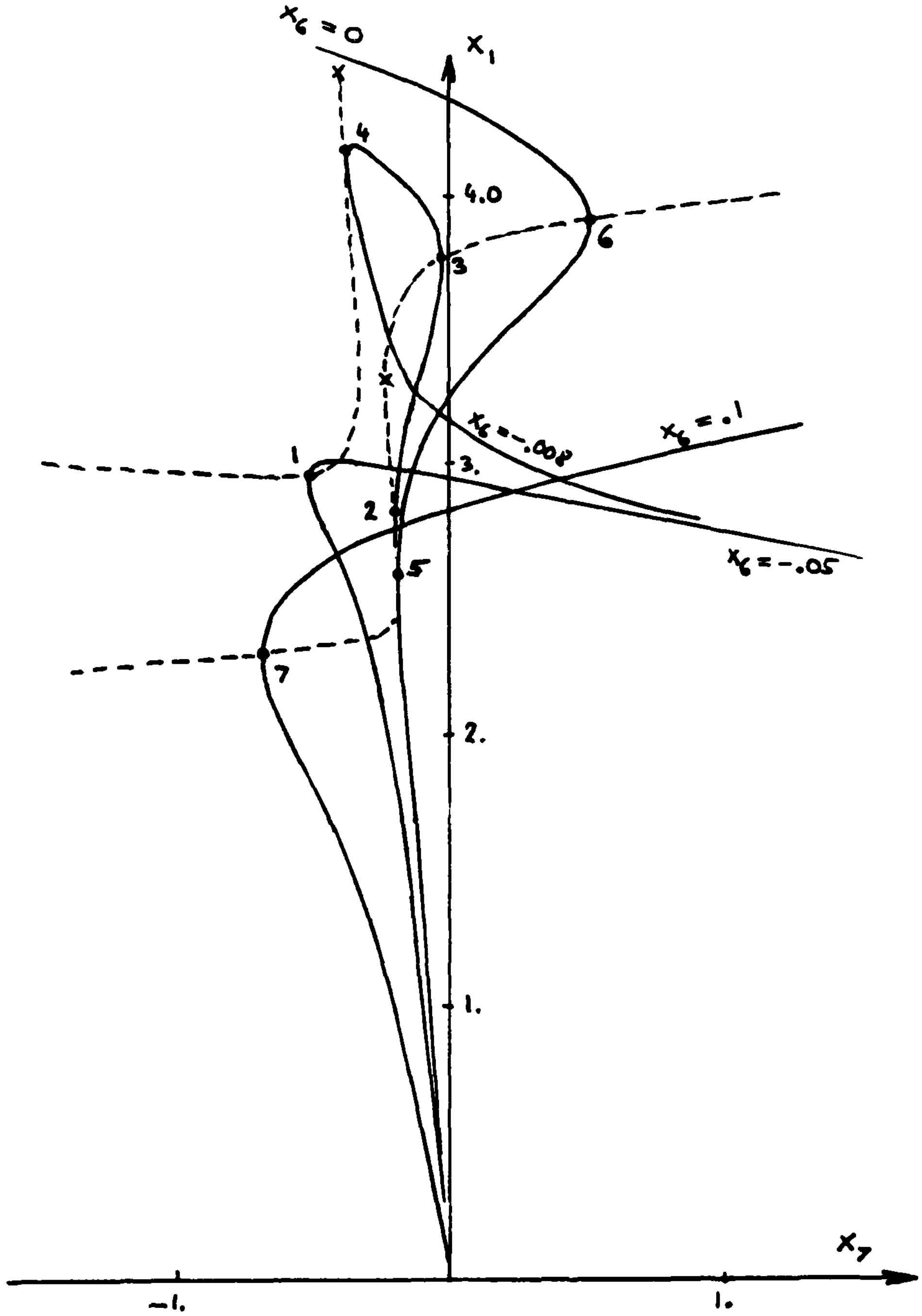


Figure 3

earlier mentioned task (1.3) (iv) (see [28]).

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
1	2.9649	0.82557	0.073661	0.041309	0.26735	-0.05	0.50481
2	2.8174	-0.17629	0.089926	0.026429	-0.071476	-0.008	-0.20497
3	3.7579	-0.65542	0.38658	0.092521	-0.19867	-0.008	0.006208
4	4.1638	0.089131	0.094806	0.022889	0.016232	-0.008	-0.37766
5	2.5873	-0.22355	0.054682	0.013676	-0.091687	0.0	-0.18691
6	3.9005	-1.1482	0.58156	0.13352	-0.32859	0.0	0.51016
7	2.2992	-1.4102	-0.061849	-0.079009	-0.58630	0.1	-0.68972
8	4.4565	-4.4909	1.6164	0.33091	-1.0857	0.1	10.0212

Table 5

Example 3. As an example for the numerical investigation of the equilibrium surface of a mechanical structure, we consider a clamped, thin, shallow, circular arch which has been used as a test case by various authors (see eg. [12], [16], [36]). Let U and W be the radial and axial displacements, R the arch radius, A the cross-sectional area, H the thickness, and E Young's modulus. With the dimensional displacements $u = U/H$, $w = W/H$, the total potential energy -- non-dimensionalized by dividing by $EAR(H/R)^2$ --

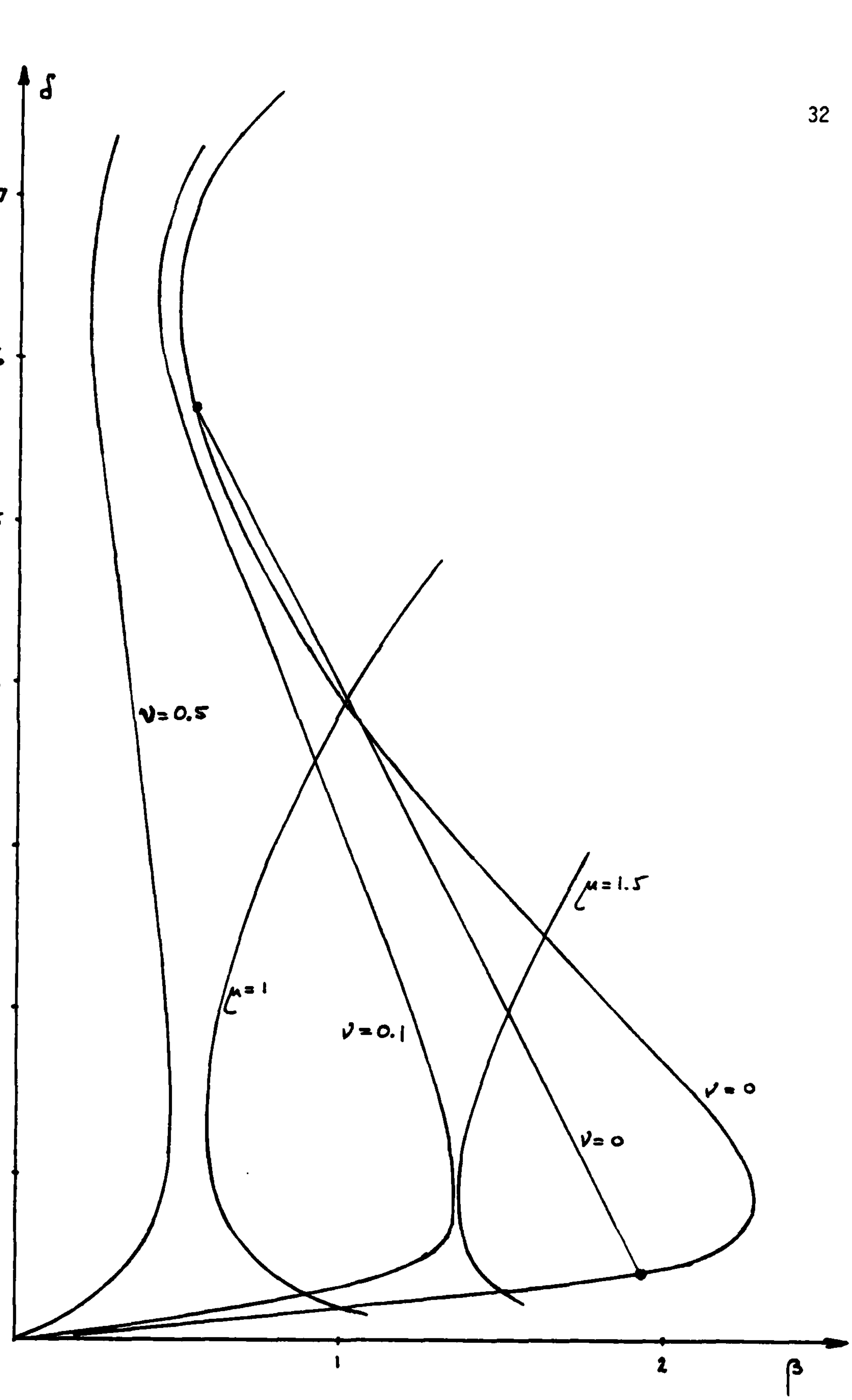


Figure 4

is given by

$$\int_{-\theta_0}^{\theta_0} \{[(\frac{dw}{d\theta} - u) + \frac{1}{2}\frac{H}{R}(\frac{du}{d\theta})^2]^2 + \alpha_1 (\frac{d^2u}{d\theta^2})^2 - \alpha_2\, p\, u\} d\theta. \tag{6.2}$$

Here $p = p(\theta)$ is the dimensionless radial load, and α_1, α_2 are dimensionless constants. Each end is assumed to be pinned, that is, we have the boundary conditions

$$u(\pm\,\theta_0) = 0, \quad w(\pm\,\theta_0) = 0, \quad \frac{d^2u}{d\theta^2}(\pm\,\theta_0) = 0. \tag{6.3}$$

The finite element approximation introduced in [36] was applied. More specifically, we used a uniform mesh with eight elements, $\theta_0 = 15$ and the constants $\alpha_1 = 3.8716 \times 10^{-6}$, $\alpha_2 = 1.65504 \times 10^{-1}$ corresponding to the data in [16]. Moreover, the following load function $p = p(\mu,\nu)$ was chosen

$$p(\mu,\nu) = \begin{cases} \mu(1 + 7\nu), & \text{for element 4} \\ \mu(1 - \nu), & \text{otherwise} \end{cases} \tag{6.4}$$

corresponding to a base load $\beta = \mu(1-\nu)$ and an excess load $8\mu\nu$ in element 4 such that the average load is always μ.

Several curves on the equilibrium surface corresponding to constant values of μ or ν were computed. Figure 4 shows the projection of these curves into the (β,δ)-plane where δ represents the radial displacement of the center point. For uniform loads, that is, $\nu = 0$, we encounter two bifurcation points on the primary curve which are connected by two "buckling" curves that have the same projection in the (β,δ)-plane.

7. THE PITCON CODE

```
      SUBROUTINE PITCON(NVAR,LIM,IT,XIT,KSTEP,IPC,H,IRET,IMOD,IPVT,
     1 HMAX,HMIN,HFACT,ABSERR,RELERR,RWORK,ISIZE)
C
C***********************************************************************
C
C   PITCON.FOR
C
C   29 OCT 1981 VERSION OF PITCON,
C   THE UNIVERSITY OF PITTSBURGH CONTINUATION PACKAGE.
C   THIS VERSION USES SINGLE PRECISION AND FULL MATRIX STORAGE.
C
C
C   THIS PACKAGE WAS PREPARED WITH THE PARTIAL SUPPORT OF
C   THE NATIONAL SCIENCE FOUNDATION, UNDER GRANT MCS-78-05299,
C
C   BY WERNER C. RHEINBOLDT AND JOHN V. BURKARDT
C   INSTITUTE FOR COMPUTATIONAL MATHEMATICS AND APPLICATIONS
C   DEPARTMENT OF MATHEMATICS AND STATISTICS,
C   UNIVERSITY OF PITTSBURGH, PITTSBURGH, PA 15261.
C
C
C   AN EARLIER VERSION OF THE PACKAGE WAS WRITTEN IN COOPERATION WITH
C
C   GEORGE D BYRNE,
C   COMPUTING TECHNOLOGY AND SERVICES
C   EXXON RESEARCH AND ENGINEERING COMPANY
C   LINDEN, NEW JERSEY, 07036
C
C
C   THIS PACKAGE COMPUTES POINTS ALONG A SOLUTION CURVE OF AN
C   UNDERDETERMINED SYSTEM OF NONLINEAR EQUATIONS OF THE FORM FX=0.
C   THE CURVE IS SPECIFIED TO PASS THROUGH A GIVEN STARTING SOLUTION
C   X OF THE SYSTEM.  HERE X DENOTES A REAL VECTOR OF NVAR
C   COMPONENTS AND FX A REAL VECTOR OF NVAR-1 COMPONENTS.
C   NORMALLY EACH CALL TO PITCON PRODUCES A NEW POINT FURTHER ALONG
C   THE SOLUTION CURVE IN A USER-SPECIFIED DIRECTION.
C     AN OPTION ALLOWS THE SEARCH FOR AND COMPUTATION OF TARGET POINTS,
C   THAT IS, SOLUTION POINTS X FOR WHICH X(IT) = XIT FOR SOME USER
C   SPECIFIED VALUES OF IT AND XIT.
C     A FURTHER OPTION ALLOWS THE SEARCH FOR AND COMPUTATION OF LIMIT
C   POINTS FOR SPECIFIED COORDINATE LIM, THAT IS, SOLUTION POINTS FOR
C   WHICH THE LIM-TH COMPONENT OF THE TANGENT VECTOR IS ZERO.
C
C
C   EXPLANATIONS OF THE ALGORITHMS USED IN THIS PACKAGE MAY
C   BE FOUND IN THE FOLLOWING REFERENCES:
C
C
C   WERNER RHEINBOLDT,
C   SOLUTION FIELD OF NONLINEAR EQUATIONS AND CONTINUATION METHODS
C   SIAM JOURNAL OF NUMERICAL ANALYSIS, 17, 1980, PP 221-237
C
C   COR DEN HEIJER AND WERNER RHEINBOLDT,
C   ON STEPLENGTH ALGORITHMS FOR A CLASS OF CONTINUATION METHODS,
C   SIAM JOURNAL OF NUMERICAL ANALYSIS 18, 1981, PP 925-947
C
C   WERNER RHEINBOLDT,
C   NUMERICAL ANALYSIS OF CONTINUATION METHODS FOR NONLINEAR
C   STRUCTURAL PROBLEMS,
C   COMPUTERS AND STRUCTURES, 13, 1981, PP 103-114
C
C
C   OUTLINE OF THE MATHEMATICAL PROCEDURE
C
C
C     THE FUNCTION F DEFINING THE SYSTEM OF EQUATIONS IS SUPPOSED TO
C   BE CONTINUOUSLY DIFFERENTIABLE.  THE REGULARITY SET R(F) OF F IS
C   THE SET OF ALL POINTS X IN NVAR-DIMENSIONAL SPACE WHERE THE
C   DERIVATIVE DF(X) HAS MAXIMAL RANK.  THE STARTING POINT IS ASSUMED
C   TO BE IN R(F).  FOR ANY INDEX IP WITH 1.LE.IP.LE.NVAR, LET
```

```
C  FA(X,IP) BE THE FUNCTION OBTAINED BY AUGMENTING F WITH
C  AN NVAR-TH SCALAR FUNCTION X(IP)-B FOR SOME NUMBER B.  FOR ANY
C  X IN R(F) THERE IS AT LEAST ONE INDEX IP SUCH THAT THE DERIVATIVE
C  DFA(X,IP) OF FA IS NONSINGULAR.  WITH SUCH AN IP, AND A DIRECTION
C  DIR (=+1 OR -1), THE TANGENT OF FA, TAN(X), IS UNIQUELY DEFINED BY:
C
C    TAN:= SOLUTION OF (  DFA(X,IP)*TAN=E(NVAR)  )
C    TAN:= TAN/(EUCLIDEAN NORM OF TAN)
C    SN:=  DIR*SIGN (DETERMINANT (DFA(X,IP) ) )
C    TAN:= SN*TAN
C
C    HERE E(I) IS THE I-TH BASIS VECTOR IN NVAR-SPACE.
C    THE PROCESS USES A LOCAL PARAMETERIZATION OF THE CURVE.
C  NORMALLY THE CONTINUATION PARAMETER INDEX IP IS CHOSEN AS
C  THAT INDEX FOR WHICH ABS(TAN(X)(IP)) IS MAXIMAL.  BUT IN THE
C  CASE OF CERTAIN CURVATURE CHANGES WHERE IT APPEARS THAT
C  A LIMIT POINT FOR THIS CHOICE FOR IP IS APPROACHING,
C  OTHER CHOICES FOR IP MAY BE USED.
C    PREDICTION TAKES PLACE ALONG THE EULER LINE X+H*TAN.  THE
C  STEPLENGTH ALGORITHM TAKES INTO ACCOUNT THE QUALITY OF THE
C  CORRECTOR ITERATION AT THE LAST POINT AND A PREDICTION OF THE
C  CHANGE IN CURVATURE.  THE TANGENTIAL STEPSIZE USED IN PREDICTION IS
C  CHOSEN SO AS TO ACHIEVE APPROXIMATELY THE PREDICTED SECANT STEPSIZE
C  AFTER CORRECTION IS DONE.
C    THE CORRECTOR ITERATION STARTS FROM THE PREDICTED POINT AND SOLVES
C  THE AUGMENTED SYSTEM FA(X,IP)=0 WITH THE VALUE OF THE SCALAR
C  B EQUAL TO THE IP-TH COMPONENT OF THE PREDICTED POINT.  THE USER
C  CAN SPECIFY AS CORRECTOR ITERATION EITHER A FULL NEWTON PROCESS,
C  OR A MODIFIED NEWTON PROCESS WITH FIXED JACOBIAN DFA EVALUATED AT
C  THE PREDICTED POINT.
C
C
C  OUTLINE OF THE COMPUTATIONAL ALGORITHM
C
C
C  DURING THE FOLLOWING DESCRIPTION, WE WILL ASSUME THAT WE
C  HAVE ENTERED THE CONTINUATION LOOP WITH AN OLD POINT XL,
C  A CURRENT POINT XC, THE TANGENT TL AT XL, AND CERTAIN SCALAR
C  QUANTITIES ASSOCIATED WITH THESE VECTORS.  WE WILL CHECK
C  FIRST FOR ANY TARGET OR LIMIT POINTS BETWEEN XL AND XC,
C  THEN PROCEED TO COMPUTE A NEW CONTINUATION POINT XF.
C  THESE NAMES ARE NOT IN PRECISE ACCORDANCE WITH THE STORAGE
C  ARRANGEMENTS UNTIL THE END OF A CONTINUATION STEP.
C
C
C  STEP 1:   FOR KSTEP.GT.0, THE CODE GOES TO STEP 2.
C            ON THE FIRST CALL TO PITCON FOR A GIVEN PROBLEM (KSTEP=-1
C            OR KSTEP=0) PROBLEM-DEPENDENT CONSTANTS ARE SET
C            AND USER CONTROL PARAMETERS ARE LOADED OR DEFAULTS USED.
C            IF (KSTEP.EQ.0), THE PROGRAM PROCEEDS TO STEP 2.
C            IF (KSTEP.EQ.-1), THE USER REQUESTS THAT THE INPUT STARTING
C            POINT XR BE CHECKED FOR THE CONDITION
C            ABS(F(XR)).LE.(ABSERR/2).  IF THIS IS NOT THE CASE, NEWTON'S
C            METHOD IS APPLIED TO THE POINT XR UNTIL THE ERROR CONDITION
C            IS SATISFIED, OR A FAILURE HAS OCCURRED.  AN UNIMPROVABLE
C            POINT RESULTS IN A RETURN OF IRET=-6.
C            IF THE STARTING POINT XR WAS IMPROVED, THE PROGRAM RETURNS
C            WITH IRET=0 AND KSTEP=0.
C            IF (KSTEP.EQ.0), THE CONTINUATION LOOP BEGINS WITH THE
C            STARTING POINT XR STORED IN XL AND XC, THE STEPSIZE HTANCF
C            SET TO THE INPUT VALUE OF H, AND THE
C            CONTINUATION PARAMETER SET TO THE INPUT VALUE OF IPC.
C            FOR KSTEP.GT.0, THESE QUANTITIES ARE COMPUTED AND
C            UPDATED BY THE PROGRAM ITSELF.
C
C  STEP 2:   TARGET POINT CHECK.  IF (IT.NE.0), A TARGET POINT IS
C            DESIRED.  THE VALUES OF XL(IT) AND XC(IT) ARE COMPARED TO
C            XIT.  IF THE TARGET POINT IS BETWEEN XL AND XC, THE PROGRAM
C            COMPUTES THE TARGET POINT, SETS IRET=1, AND
C            RETURNS, TEMPORARILY INTERRUPTING NORMAL CONTINUATION.
```

```
C
C  STEP 3:  TANGENT AND LOCAL CONTINUATION PARAMETER CALCULATION. IF THE
C           LOOP WAS SUSPENDED AT THE LAST CALL TO PITCON TO ALLOW THE
C           RETURN OF A LIMIT POINT, THEN THE TANGENT HAS ALREADY BEEN
C           CALCULATED AND A LIMIT POINT CHECK IS SUPERFLUOUS, SO
C           THE PROGRAM SKIPS TO STEP 5.
C           OTHERWISE, A VECTOR IN THE TANGENT PLANE AT XC IS COMPUTED.
C           SUPPOSE THAT THE PREVIOUS CONTINUATION PARAMETER INDEX
C           WAS IPL, WHERE ON THE FIRST STEP IPL IS USER-SUPPLIED.
C           THE NEW TANGENT IS NORMALIZED, AND THE IPL-TH COMPONENT
C           IS FORCED TO HAVE THE SAME SIGN AS THE IPL-TH COMPONENT
C           OF THE PREVIOUS TANGENT (OR ON FIRST STEP, THE SAME
C           SIGN AS THE USER INPUT DIRECTION DIR.)  THEN THE LOCAL
C           CONTINUATION PARAMETER IPC IS DETERMINED.  IPC IS SET TO THE
C           LOCATION OF THE LARGEST COMPONENT OF THE TANGENT VECTOR
C           UNLESS A LIMIT POINT FOR THIS CHOICE APPEARS TO BE
C           APPROACHING, IN WHICH CASE THE LOCATION OF THE SECOND
C           LARGEST COMPONENT MAY BE TRIED.
C           ONCE IPC IS SET, THE QUANTITIES TCIPC, TCLIM, HSECLC, ALPHLC
C           AND DIR ARE COMPUTED, WHOSE MEANINGS ARE EXPLAINED BELOW.
C
C  STEP 4:  LIMIT POINT CHECK.  IF LIM.NE.0, THE LIM-TH COMPONENTS
C           OF THE OLD AND NEW TANGENTS ARE COMPARED.  IF THESE DIFFER
C           IN SIGN, A LIMIT POINT LIES BETWEEN XL AND XC.  THE PROGRAM
C           ATTEMPTS TO FIND THIS LIMIT POINT.  IF FOUND, IT STORES
C           THE LIMIT POINT IN XR, THE TANGENT AT XR IN TL, SETS IRET=2,
C           AND RETURNS, TEMPORARILY INTERRUPTING THE NORMAL LOOP.
C
C  STEP 5:  STEP LENGTH COMPUTATION.  THE PROGRAM COMPUTES HTANCF, THE
C           STEPSIZE TO BE USED ALONG THE TANGENT TO OBTAIN THE
C           PREDICTED POINT XPRED:=XC+HTANCF*TC, THE STARTING POINT
C           FOR THE CORRECTOR PROCESS.  IN COMPUTING HTANCF, CERTAIN
C           CURVATURE AND STEPSIZE DATA ARE UPDATED.
C
C  STEP 6:  PREDICTION AND CORRECTION STEP.  WITH THE PREDICTED POINT
C           XPRED=XC+HTANCF*TC AS A STARTING POINT, THE CORRECTOR
C           PROCESS IS APPLIED TO CORRECT THE POINT UNTIL
C           ABS(F(XCOR)).LE.ABSERR AND XSTEP, THE LAST CORRECTOR STEP,
C           SATISFIES XSTEP.LE.ABSERR+RELERR*ABS(XCOR).
C           IF THE SIZE OF A CORRECTOR STEP IS TOO LARGE,
C           OR IF A CORRECTION STEP INCREASES THE FUNCTION VALUE, OR
C           THE MAXIMUM NUMBER OF STEPS ARE TAKEN WITHOUT CONVERGENCE,
C           THE STEPSIZE HTANCF IS REDUCED AND THE CORRECTOR STEP IS
C           ATTEMPTED AGAIN.  IF THE STEPSIZE SHRINKS BELOW HMIN, THE
C           PROGRAM SETS AN ERROR FLAG AND RETURNS.
C
C  STEP 7:  STORING INFORMATION BEFORE RETURN.  AFTER A SUCCESSFUL
C           CONTINUATION STEP, THE PROGRAM REARRANGES ITS STORAGE SO
C           THAT THE ENTRIES CORRESPONDING TO XC AND XF HOLD THE PROPER
C           DATA, COMPUTES CORDIS, THE SIZE OF THE CORRECTION TO THE
C           PREDICTED POINT, AND MODIFIES CORDIS TO A VALUE THAT WOULD
C           CORRESPOND TO AN OPTIMAL NUMBER OF CORRECTOR STEPS.
C
C
C  ON NORMAL RETURN, THE VECTOR XR (THE FIRST NVAR ENTRIES OF RWORK),
C  CONTAINS A SOLUTION POINT ON THE CURVE F(XR)=0, AND IS EITHER
C  A CONTINUATION POINT, A TARGET POINT, OR A LIMIT POINT, WHICH
C  IS INDICATED BY THE VALUE OF IRET.
C  IF IRET IS NEGATIVE, AN ERROR HAS OCCURRED.  IF A LIMIT POINT IS
C  RETURNED, THE TANGENT VECTOR AT THE LIMIT POINT IS CONTAINED IN THE
C  LOCATION TL IN RWORK.  ON FIRST CALL, THE USER MUST SET SOME OF THE
C  SCALAR PARAMETERS, AND THE STARTING POINT XR.  THEREAFTER, ONLY IT
C  AND XIT SHOULD BE CHANGED BY THE USER DURING A PROBLEM RUN.
C    IF A NEW PROBLEM IS TO BE RUN (WHETHER A DIFFERENT FUNCTION, OR THE
C  SAME FUNCTION WITH DIFFERENT STARTING POINT OR ERROR CONTROLS)
C  THE PROGRAM MAY BE RESET BY USING KSTEP=-1 OR 0, AT WHICH TIME THE
C  SCALARS AND THE POINT XR MUST BE SET AGAIN.  NOTE THAT IN THIS CASE
C  THE STATISTICAL DATA IN THE COMMON BLOCKS /COUNT1/ AND /COUNT2/
C  WILL BE RESET TO 0 AS WELL.
C
```

```
C
C  DEFINITIONS AND DEFAULTS OF PITCON PARAMETERS
C
C
C  NVAR  = THE NUMBER OF VARIABLES IN THE NONLINEAR SYSTEM.  NVAR IS
C          THE DIMENSION OF THE PIVOT VECTOR IPVT, AND THE SIZE OF THE
C          VECTORS XR, XC, XF, TL AND TC WHICH ARE CONTAINED IN RWORK.
C          RWORK ALSO CONTAINS STORAGE FOR THE MATRIX FPRYM WHICH
C          IS OF SIZE NVAR X NVAR.
C          NVAR MUST BE GREATER THAN 1, AND MUST NOT BE CHANGED DURING
C          THE COURSE OF A PROBLEM RUN.  NVAR HAS NO DEFAULT VALUE.
C  LIM   = LIMIT POINT FLAG AND INDEX.  IF (LIM.EQ.0), LIMIT POINTS
C          ARE NOT TO BE LOOKED FOR.  OTHERWISE, THE USER SHOULD SET
C          LIM TO THE INDEX OF THE VARIABLE FOR WHICH LIMIT
C          POINTS ARE TO BE SOUGHT.  LIM DEFAULTS TO ZERO.
C          LIM MUST SATISFY 0.LE.LIM.LE.NVAR.
C  IT    = TARGET POINT FLAG AND INDEX.  IF (IT.EQ.0), TARGET POINTS
C          ARE NOT TO BE LOOKED FOR.  OTHERWISE, THE USER SHOULD SET
C          IT TO THE INDEX OF THE VARIABLE FOR WHICH TARGET
C          VALUES XIT ARE DESIRED.  IT HAS THE DEFAULT VALUE ZERO.
C          IT MAY BE RESET BY THE USER AT ANY TIME DURING A RUN.
C          IT MUST SATISFY 0.LE.IT.LE.NVAR.
C  XIT   = THE VALUE OF THE TARGET VECTOR COMPONENT SOUGHT, IF IT.NE.0.
C          TARGET POINTS XR SATISFY XR(IT)=XIT.  XIT HAS NO DEFAULT.
C  KSTEP = THE NUMBER OF CONTINUATION STEPS TAKEN.  THIS DOES NOT
C          INCLUDE FAILURES, TARGET POINTS OR LIMIT POINTS.  THE PROGRAM
C          INCREMENTS KSTEP EACH TIME A NEW POINT XF IS COMPUTED.
C          ON THE FIRST CALL TO PITCON FOR A PARTICULAR PROBLEM, THE
C          USER SHOULD SET KSTEP TO 0 OR -1.  IF KSTEP=-1, THE PROGRAM
C          WILL CHECK THE ACCURACY OF THE STARTING POINT XR, AND IF
C          NECESSARY, ATTEMPT TO CORRECT IT USING NEWTON'S METHOD.
C          IF KSTEP=0, THE PROGRAM PERFORMS NO CHECK ON THE STARTING
C          POINT, AND PROCEEDS TO THE CONTINUATION LOOP.  IF THE USER
C          WISHES TO RUN A DIFFERENT PROBLEM THEN A CALL TO PITCON
C          WITH KSTEP=-1 OR 0 WILL RESET THE CODE, DESTROYING THE
C          INFORMATION FROM THE PREVIOUS RUN.  KSTEP DEFAULTS TO -1.
C  IPC   = THE COMPONENT OF XC TO BE USED AS CONTINUATION PARAMETER.
C          ON THE FIRST CALL ONLY, THE USER OUGHT TO SET IPC OR ALLOW
C          THE DEFAULT VALUE IPC=NVAR.  AFTER THE FIRST CALL, THE
C          DETERMINATION OF IPC IS DONE BY THE PROGRAM USING INFORMATION
C          ABOUT THE TANGENT VECTOR AT XC.
C  H     = SUGGESTED STARTING STEP SIZE ALONG THE TANGENT TO THE CURVE.
C          IF H=0.0 ON THE INITIAL CALL, H DEFAULTS TO (HMAX+HMIN)/2
C          IF H IS NEGATIVE ON THE FIRST CALL, THE MINUS SIGN
C          IS ABSORBED BY DIR AND INDICATES THAT THE DIRECTION OF THE
C          FIRST STEP SHOULD BE IN THE NEGATIVE IPC DIRECTION.
C          AFTER THE FIRST STEP, STEPSIZE IS CONTROLLED BY THE PROGRAM.
C          UPON RETURN WITH A CONTINUATION POINT (IRET=0), H IS
C          OVERWRITTEN BY HTANCF, THE STEPSIZE USED IN REACHING THE
C          NEW POINT.
C  IRET  = A RETURN FLAG TO INDICATE ERRORS OR THE TYPE OF POINT
C          RETURNED IN XR.  NONNEGATIVE VALUES OF IRET REPRESENT
C          NORMAL RETURNS.  NEGATIVE VALUES OF IRET INDICATE THAT
C          SOME ERROR OR DIFFICULTY HAS BEEN ENCOUNTERED.  VALUES
C          OF IRET BETWEEN -1 AND -4 ARE SIMPLY REPORTS THAT AN
C          ATTEMPT TO COMPUTE A LIMIT OR TARGET POINT FAILED.  THESE
C          DO NOT AFFECT FURTHER CONTINUATION STEPS, AND THE USER NEED
C          NOT MODIFY ANY VARIABLES BEFORE PROCEEDING.
C          VALUES OF IRET OF -5 AND -6 REFER TO DANGEROUS SITUATIONS
C          THAT MAY BE CORRECTABLE.
C          VALUES OF IRET FROM -7 TO -10 ARE SERIOUS, FATAL ERRORS.
C          THE USER SHOULD HALT THE PROGRAM AND EXAMINE HIS INPUT
C          AND THE INTERIM RESULTS.
C          IRET SHOULD BE ZERO ON THE FIRST CALL FOR A PROBLEM.
C          THE SPECIFIC VALUES OF IRET AND THEIR MEANINGS ARE:
C
C          IRET=2:   NORMAL RETURN WITH LIMIT POINT IN XR AND TANGENT
C                    AT XR CONTAINED IN TL.
C
C          IRET=1:   NORMAL RETURN WITH TARGET POINT IN XR.
```

```
C
C         IRET=0:   NORMAL RETURN WITH NEW CONTINUATION POINT IN XR.
C
C         IRET=-1:  AN ERROR OCCURRED DURING COMPUTATION OF LIMIT POINT
C
C         IRET=-2:  CORECT CALLED FOR TARGET POINT CALCULATION FAILED
C                   AFTER KNMAX ITERATIONS.
C
C         IRET=-3:  SOLVE WAS CALLED BY CORECT FOR TARGET POINT
C                   CALCULATION, AND FAILED.  (MATRIX ELIMINATION FOUND
C                   ZERO PIVOT).
C
C         IRET=-4:  UNACCEPTABLE CORRECTOR STEP IN TARGET POINT
C                   CALCULATION.
C
C         IRET=-5:  PREDICTION STEP HTANCF IS LESS THAN HMIN, PERHAPS
C                   BECAUSE OF REPEATED FAILURE OF CORECT, AND
C                   CONSEQUENT STEPSIZE REDUCTION.  USER MIGHT REDUCE
C                   HMIN, OR SWITCH FROM IMOD=1 TO IMOD=0, OR INCREASE
C                   ABSERR AND RELERR.  BUT BE AWARE THAT REPEATED
C                   STEPSIZE REDUCTIONS MAY INDICATE AN INTRACTABLE
C                   FUNCTION.
C
C         IRET=-6:  FUNCTION VALUE FNRMXF OF INPUT XR IS TOO LARGE
C                   AND COULD NOT BE IMPROVED BY CORECT.  USER
C                   MIGHT RECOVER BY RELAXING ERROR CONTROLS, IMPROVING
C                   STARTING POINT XR, OR CHANGING VALUE OF IPC.
C
C         IRET=-7:  SOLVE FAILED IN A CALL FROM TANGNT.
C
C         IRET=-8:  SOLVE FAILED IN A CALL FROM CORECT.
C
C         IRET=-9:  THE TANGENT VECTOR TC AT XC IS ZERO.
C
C         IRET=-10: IMPROPER INPUT, NVAR.LE.1, OR
C                   ISIZE.LT.(NVAR)*(NVAR+5), OR
C                   PROGRAM HAS BEEN CALLED AGAIN AFTER FATAL ERROR.
C
C  IMOD   = METHOD FLAG FOR CORRECTOR STEP, SPECIFYING TYPE OF
C           NEWTON METHOD TO BE USED.
C
C         IMOD=0: UPDATE JACOBIAN FOR TANGENT CALCULATION,
C                 UPDATE JACOBIAN FOR EACH CORRECTOR STEP.
C
C         IMOD=1: UPDATE JACOBIAN FOR TANGENT CALCULATION,
C                 EVALUATE JACOBIAN AT FIRST CORRECTOR STEP ONLY.
C
C  IPVT   = INTEGER VECTOR USER DECLARED TO BE OF SIZE NVAR.
C           USED DURING THE MATRIX FACTORIZATION TO STORE PIVOT
C           INFORMATION.
C  HMAX   = THE MAXIMUM STEP SIZE. IF HMAX.LE.0.0 ON INITIAL CALL,
C           HMAX DEFAULTS TO SQRT(NVAR).
C  HMIN   = THE MINIMUM STEP SIZE. IF HMIN.LE.SQRT(EPMACH) ON INITIAL
C           CALL, HMIN DEFAULTS TO SQRT(EPMACH), WHERE EPMACH IS THE
C           MACHINE PRECISION CONSTANT.
C  HFACT  = LIMIT ON STEPSIZE CHANGE.  HSECLC IS THE SECANT STEPSIZE
C           OF THE LAST STEP, AND HTANCF IS THE STEPSIZE TO BE USED
C           IN OBTAINING THE PREDICTED POINT.  THE FOLLOWING RELATIONSHIP
C           MUST BE SATISFIED: (HSECLC/HFACT).LE.HTANCF.LE.(HSECLC*HFACT)
C           IF THE CORRECTOR STEP FAILS, THEN HFACT IS ALSO USED TO
C           REDUCE THE PREDICTOR STEP HTANCF TO (HTANCF/HFACT)
C           IF HFACT.LE.1.0 ON INITIAL CALL, HFACT DEFAULTS TO 3.0.
C  ABSERR= ABSOLUTE ERROR CONTROL. IF ABSERR=0.0 ON INITIAL CALL
C           ABSERR DEFAULTS TO SQRT(EPMACH)
C  RELERR= RELATIVE ERROR CONTROL. IF RELERR=0.0 ON INITIAL CALL
C           RELERR DEFAULTS TO SQRT(EPMACH)
C  RWORK  = USER DECLARED VECTOR OF SIZE ISIZE=NVAR*(NVAR+5).
C           RWORK STORES FIVE VECTORS AND AN ARRAY IN THE ORDER
C           (XR,XC,XF,TL,TC,FPRYM).  THEIR BEGINNING LOCATIONS
C           ARE IXR=1, IXC=NVAR+1, IXF=2*NVAR+1, ITL=3*NVAR+1,
C           ITC=4*NVAR+1, IFP=5*NVAR+1.  FPRYM IS AN ARRAY OF
```

```
C           SIZE NVAR X NVAR.  THE MEANINGS OF THESE COMPONENTS OF
C           RWORK ARE DESCRIBED BELOW.  THE USER SHOULD SET A VALUE TO XR
C           ON FIRST CALL, BUT NO OTHER PORTIONS OF RWORK SHOULD BE
C           SET.  AFTER THE FIRST CALL FOR A PROBLEM, NO ENTRIES OF RWORK
C           SHOULD BE ALTERED.
C (XR)    = ON FIRST CALL, A USER SUPPLIED STARTING POINT, WHICH MAY BE
C           IMPROVED BY THE PROGRAM IF KSTEP=-1.  ON NORMAL RETURN FROM
C           PITCON, XR WILL HOLD THE MOST RECENTLY FOUND POINT, WHETHER
C           A CONTINUATION POINT, TARGET POINT, OR LIMIT POINT.
C (XC)    = THE PREVIOUS CONTINUATION POINT.
C (XF)    = THE CURRENT CONTINUATION POINT.
C (TL)    = PREVIOUS VALUE OF TANGENT VECTOR.  NOTE THAT THIS CORRESPONDS
C           TO A POINT XL WHICH HAS BEEN DISCARDED.  ON A LIMIT POINT
C           RETURN, TL WILL CONTAIN INSTEAD THE TANGENT AT THE
C           LIMIT POINT.  ON A TARGET POINT RETURN, TL WILL HAVE
C           BEEN OVERWRITTEN BY THE FUNCTION VALUE AT THE TARGET POINT.
C (TC)    = THE TANGENT VECTOR AT THE PREVIOUS CONTINUATION POINT.
C (FPRYM)= MATRIX STORAGE AREA FOR SETTING UP AND SOLVING THE
C           LINEAR SYSTEMS INVOLVING DFA(X,IP).
C  ISIZE = USER SET DIMENSION FOR VECTOR RWORK, WHICH MUST BE AT
C           LEAST OF SIZE NVAR*(NVAR+5).
C
C
C  NOMENCLATURE FOR STEP DEPENDENT VARIABLES
C
C
C  THE PROGRAM ACCUMULATES INFORMATION THAT IS ASSOCIATED WITH
C  SEVERAL PREVIOUS CONTINUATION POINTS OR THE STEPS MADE BETWEEN
C  THEM.  IN INTERPRETING THE CODE OR ITS OUTPUT, IT IS IMPORTANT
C  TO KNOW WHERE SUCH QUANTITIES APPLY.  THE FOLLOWING DESCRIPTION
C  OF SOME OF THE VARIABLES IS VALID ONLY UPON A NORMAL RETURN WITH
C  A CONTINUATION POINT.
C  THE POINTS 'XLL' AND 'XL' WILL HAVE BEEN DISCARDED BY THE PROGRAM,
C  BUT SOME QUANTITIES ASSOCIATED WITH THEM STILL SURVIVE.
C
C  QUANTITIES ASSOCIATED WITH STEP FROM 'XLL' TO 'XL':
C
C  HSECLL = SIZE OF SECANT FROM 'XLL' TO 'XL', EUCLIDEAN NORM(XLL-XL)
C
C  QUANTITIES ASSOCIATED WITH THE POINT 'XL':
C
C  IPL    = THE LOCATION OF THE FIRST OR SECOND LARGEST COMPONENT
C            OF THE TANGENT VECTOR AT 'XL'.
C  TLLIM  = VALUE OF LIM-TH COMPONENT OF TANGENT VECTOR AT 'XL'.
C  TL     = TANGENT VECTOR AT 'XL', ALTHOUGH LIMIT OR TARGET POINT
C            CALCULATIONS COULD HAVE OVERWRITTEN THIS VECTOR.
C
C  QUANTITIES ASSOCIATED WITH INTERVAL FROM 'XL' TO XC:
C
C  ALPHLC = THE ANGLE BETWEEN THE TANGENTS AT 'XL' AND XC.
C  CURVLC = ESTIMATED CURVATURE BETWEEN 'XL' AND XC.
C  HSECLC = SIZE OF SECANT BETWEEN 'XL' AND XC, EUCLIDEAN NORM(XL-XC)
C
C  QUANTITIES ASSOCIATED WITH THE POINT XC:
C
C  DETA   = BINARY MANTISSA OF DETERMINANT OF DFA(XC,IPL), DIVIDED
C            BY IPL-TH COMPONENT OF TANGENT AT XC.
C  DIR    = SIGN OF DETA, DETERMINES SENSE OF CONTINUATION.
C  IEXP   = BINARY EXPONENT OF DETERMINANT OF DFA(XC,IPL).
C  IPC    = LOCATION OF FIRST OR SECOND LARGEST COMPONENT OF TANGENT
C            VECTOR AT XC.
C  TC     = TANGENT VECTOR AT XC.
C  TCLIM  = VALUE OF LIM-TH COMPONENT OF TANGENT AT XC.
C  TCIPC  = VALUE OF TC(IPC)
C
C  QUANTITIES ASSOCIATED WITH THE INTERVAL FROM XC TO XF:
C
C  CURVCF = ESTIMATED CURVATURE BETWEEN XC AND XF.
C  HTANCF = STEPSIZE USED ALONG TANGENT TO GET PREDICTED POINT
```

```
C          WHICH WAS CORRECTED TO SOLUTION POINT XF.
C
C  QUANTITIES ASSOCIATED WITH THE POINT XF:
C
C  CORDIS = SIZE OF THE TOTAL CORRECTION FROM PREDICTED POINT
C           X=XC+HTANCF*TC TO CORRECTED POINT XF.
C           NOTE THAT THIS HAS BEEN MODIFIED TO AN 'OPTIMAL' VALUE.
C  CURVXF = A PREDICTED VALUE OF THE CURVATURE AT XF.
C  FNRMXF = MAXIMUM NORM OF FUNCTION VALUE AT XF.
C  FPRYM  = DFA(XF,IPC) HAS ACTUALLY BEEN LAST EVALUATED AT THE
C           PENULTIMATE CORRECTOR ITERATE (IF IMOD.NE.1).  IT WILL BE
C           EVALUATED AT XF AS SOON AS THE NEXT LOOP BEGINS AND THE
C           TANGENT IS NEEDED.
C  XSTEP  = SIZE OF THE LAST CORRECTOR STEP TAKEN IN CONVERGING TO XF.
C
C
C  SUBROUTINES IN THIS PACKAGE
C
C
C  PITCON(NVAR,LIM,IT,XIT,KSTEP,IPC,H,IRET,IMOD,IPVT,
C  HMAX,HMIN,HFACT,ABSERR,RELERR,RWORK,ISIZE)
C
C  DRIVING ROUTINE OF CONTINUATION CODE.  INITIALIZES INFORMATION,
C  DETERMINES WHETHER LIMIT, TARGET OR CONTINUATION POINT WILL
C  BE SOUGHT THIS STEP, COMPUTES STEPLENGTHS, CONTROLS CORRECTOR
C  PROCESS, AND HANDLES ERROR RETURNS.
C
C  CORECT(NVAR,X,IHOLD,WORK,IERR,IMOD,FPRYM,IPVT,ABSERR,RELERR,
C  XSTEP,NEQN,FNRM)
C
C  USES A FORM OF NEWTON'S METHOD TO SOLVE THE AUGMENTED NONLINEAR
C  SYSTEM FA(X)=0 WITH AUGMENTING EQUATION X(IHOLD)=B, THAT IS, X(IHOLD)
C  IS HELD FIXED DURING THE CORRECTION PROCESS.
C
C  TANGNT(NVAR,XC,IPC,TC,IRET,ICALL,FPRYM,IPVT,NEQN,DETA,IEXP)
C
C  APPLIES ALGORITHM DESCRIBED ABOVE TO SOLVE DFA(XC,IPL)*TC=E(NVAR)
C  AND THEN NORMALIZES TANGENT VECTOR, CORRECTS SIGN, AND SETS
C  IPC AND DIR.
C
C  ROOT(A,FA,B,FB,C,FC,KOUNT,IFLAG)
C
C  ROOT FINDER USED TO LOCATE LIMIT POINT.  THIS ROUTINE IS A MODIFIED
C  VERSION OF THE FORTRAN FUNCTION ZERO GIVEN IN THE BOOK:
C  'ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES'
C  BY RICHARD P BRENT, PRENTICE HALL, 1973.
C
C  SOLVE(NVAR,X,Y,IP,DETA,IEXP,IERR,ICALL,IMOD,FPRYM,IPVT)
C
C  SETS UP AND SOLVES THE SYSTEM DFA(X,IP)*Y(OUTPUT)=Y(INPUT)
C  WHERE DFA(X,IP) IS THE JACOBIAN OF FA AT X,
C  AND Y IS A RIGHT HAND SIDE SUPPLIED BY THE CALLING ROUTINE.
C
C  **NOTE** SUBROUTINE SOLVE USES FULL MATRIX STORAGE TO SOLVE THE
C  SYSTEM.  THE USER MAY WISH TO REPLACE THIS ROUTINE WITH ONE MORE
C  SUITED TO HIS PROBLEM.
C
C
C  USER SUPPLIED SUBROUTINES
C
C
C  FCTN(NVAR,X,FX)
C
C  EVALUATES THE NVAR-1 COMPONENT FUNCTION FX GIVEN X AN NVAR COMPONENT
C  VECTOR. THIS FUNCTION DESCRIBES THE NONLINEAR SYSTEM.  THE AUGMENTING
C  EQUATION IS HANDLED BY THE CONTINUATION PACKAGE.
C
C  FPRIME(NVAR,X,FPRYM)
C
C  EVALUATES THE NVAR-1 BY NVAR JACOBIAN MATRIX FPRYM (X)
```

```
C     AT X AND STORES IT IN THE NVAR BY NVAR ARRAY FPRYM,
C     SO THAT  FPRYM(I,J) CONTAINS (DF(X) (I)/ DX (J)).
C     THE LAST ROW OF FPRYM (FOR THE AUGMENTING EQUATION) IS INSERTED
C     BY THE ROUTINE SOLVE.
C
C
C     LINPAK ROUTINES USED
C
C
C     LINPAK REFERENCE:
C     LINPACK USER'S GUIDE,
C     J J DONGARRA, J R BUNCH, C B MOLER AND G W STEWART,
C     SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS,
C     PHILADELPHIA, 1979.
C
C
C     ISAMAX(N,SX,INCX)
C     INTEGER FUNCTION RETURNS THE POSITION OF LARGEST ELEMENT OF SX
C
C     SAXPY(N,SA,SX,INCX,SY,INCY)
C     SETS VECTOR SY(I) = SA*SX(I)+SY(I)
C
C     SCOPY(N,SX,INCX,SY,INCY)
C     SETS SY(I)=SX(I)
C
C     SDOT(N,SX,INCX,SY,INCY)
C     SDOT = SUM(I=1 TO N) SX(I)*SY(I)
C
C     SNRM2(N,SX,INCX)
C     SNRM2 = EUCLIDEAN NORM OF SX(I)
C
C     **NOTE**  SNRM2 HAS MACHINE DEPENDENT CUTOFF CONSTANTS
C
C     SSCAL(N,SA,SX,INCX)
C     SETS SX(I)=SA*SX(I)
C
C     SGEFA(A,LDA,N,IPVT,INFO)
C     FACTORS MATRIX A WHOSE LEADING DIMENSION WAS DECLARED AS LDA
C     AND WHOSE ACTUAL USED DIMENSION IS N, SETS UP PIVOT INFORMATION
C     IN VECTOR IPVT AND WARNS OF ZERO PIVOTS.
C
C     SGESL(A,LDA,N,IPVT,B,JOB)
C     ACCEPTS OUTPUT OF SGEFA, AND A RIGHT HAND SIDE B, AND SOLVES
C     SYSTEM A*X=B, RETURNING X IN B.  FOR MODIFIED NEWTON'S METHOD, ONCE
C     MATRIX IS FACTORED BY SGEFA, ONLY SGESL IS CALLED FOR SUCCESSIVE
C     RIGHT HAND SIDES
C
C
C
C     LABELED COMMON BLOCKS
C
C
C     /COUNT1/  COUNTS NUMBER OF CALLS FROM ... TO ... AS FOLLOWS:
C     ICRSL  = CORECT TO SOLVE
C     ITNSL  = TANGNT TO SOLVE
C     NSTCR  = PITCON TO CORECT FOR IMPROVED STARTING POINT
C     NCNCR  = PITCON TO CORECT FOR CONTINUATION POINT
C     NTRCR  = PITCON TO CORECT FOR TARGET POINTS
C     NLMCR  = PITCON TO CORECT FOR LIMIT POINT
C     NLMRT  = PITCON TO ROOT FOR LIMIT POINT
C
C     NOTE THAT NSTCR, NCNCR, NTRCR, NLMCR AND NLMRT COUNT THE NUMBER
C     OF ITERATIVE STEPS (NEWTON OR ROOTFINDING) AND NOT JUST THE NUMBER
C     OF SUBROUTINE CALLS.
C
C     /COUNT2/  KEEPS PERFORMANCE AND WORK STATISTICS
C     IFEVAL = NUMBER OF CALLS TO FCTN
C     IPEVAL = NUMBER OF CALLS TO FPRIME
C     ISOLVE = NUMBER OF CALLS TO SOLVE
C     NRED   = NUMBER OF STEPSIZE REDUCTIONS MADE BEFORE PREDICTOR
```

```
C             POINT CONVERGED TO THE NEW CONTINUATION POINT.                   PTCN0561
C   NRDSUM = TOTAL NUMBER OF STEPSIZE REDUCTIONS                                PTCN0562
C   KN     = NUMBER OF CORRECTOR ITERATION STEPS TAKEN IN MOST RECENT           PTCN0563
C            CALL TO CORECT.                                                    PTCN0564
C   KNSUM  = TOTAL NUMBER OF CORRECTOR ITERATION STEPS.                         PTCN0565
C                                                                               PTCN0566
C   /OUTPUT/                                                                    PTCN0567
C   IWRITE = USER ACCESSIBLE OUTPUT INDICATOR.                                  PTCN0568
C            IWRITE=0, NO OUTPUT PRINTED BY PITCON.                             PTCN0569
C            IWRITE=1, ERROR MESSAGES PRINTED BY PITCON.                        PTCN0570
C            IWRITE=2, CERTAIN OUTPUT WILL BE PRINTED BY PITCON.                PTCN0571
C                                                                               PTCN0572
C   /POINT/ CONTAINS DATA ABOUT THE SOLUTION CURVE                              PTCN0573
C   DETA   = BINARY MANTISSA OF THE DETERMINANT OF THE AUGMENTED JACOBIANPTCN0574
C   IEXP   = BINARY EXPONENT OF THE DETERMINANT OF THE AUGMENTED JACOBIANPTCN0575
C   CURVCF = ESTIMATED CURVATURE BETWEEN XC AND XF.                             PTCN0576
C   CORDIS = NORM OF THE CORRECTOR STEP FROM PREDICTED POINT TO CORRECTEDPTCN0577
C            POINT, USING MAXIMUM ABSOLUTE VALUE AS THE NORM.                   PTCN0578
C            THIS QUANTITY IS MODIFIED TO AN 'OPTIMAL' VALUE.                   PTCN0579
C   ALPHLC = ANGLE BETWEEN OLD AND NEW TANGENTS TL AND TC                       PTCN0580
C   HSECLC = EUCLIDEAN NORM OF SECANT BETWEEN XL AND XC.                        PTCN0581
C   FNRMXF = MAXIMUM NORM OF FUNCTION VALUE AT NEW CONTINUATION POINT.          PTCN0582
C                                                                               PTCN0583
C   /TOL/                                                                       PTCN0584
C   EPMACH= SMALLEST NUMBER SUCH THAT 1.0+EPMACH.GT.EPMACH                      PTCN0585
C           .5*BETA**(1-TAU) FOR ROUNDED, TAU-DIGIT ARITHMETIC                  PTCN0586
C           BASE BETA.  TWICE THIS VALUE FOR TRUNCATED ARITHMETIC.              PTCN0587
C           THIS IS THE RELATIVE MACHINE PRECISION.                             PTCN0588
C           EPMACH=2**(-27) FOR DEC-10.                                         PTCN0589
C   EPSATE= 8*EPMACH                                                            PTCN0590
C   EPSQRT= SQUARE ROOT OF EPMACH                                               PTCN0591
C                                                                               PTCN0592
C                                                                               PTCN0593
C   PROGRAMMING NOTES                                                           PTCN0594
C                                                                               PTCN0595
C                                                                               PTCN0596
C   THE USER MUST -                                                             PTCN0597
C                                                                               PTCN0598
C   1.  WRITE SUBROUTINES                                                       PTCN0599
C   SUPPLY A CALLING PROGRAM, AND THE TWO ROUTINES FCTN AND FPRIME              PTCN0600
C   AS DESCRIBED ABOVE.                                                         PTCN0601
C                                                                               PTCN0602
C   2.  SET STORAGE AREAS                                                       PTCN0603
C   DECLARE A REAL VECTOR RWORK OF SIZE ISIZE, ISIZE.GE.NVAR*(NVAR+5)           PTCN0604
C   AND AN INTEGER VECTOR IPVT OF SIZE NVAR.                                    PTCN0605
C                                                                               PTCN0606
C   3.  PASS CERTAIN NON-DEFAULTABLE VALUES                                     PTCN0607
C   PASS NVAR GREATER THAN ZERO, ISIZE.GE.NVAR*(NVAR+5)                         PTCN0608
C   AND SET IRET=0, KSTEP=-1 OR KSTEP=0 ON FIRST CALL                           PTCN0609
C   FOR A NEW PROBLEM.                                                          PTCN0610
C                                                                               PTCN0611
C   THE USER SHOULD -                                                           PTCN0612
C                                                                               PTCN0613
C   1.  STORE A STARTING POINT XR IN THE FIRST NVAR LOCATIONS OF RWORK          PTCN0614
C   BEFORE CALLING PITCON.                                                      PTCN0615
C   IF SUCH A VALUE IS NOT GIVEN, THE CODE MAY BE UNABLE TO PRODUCE ONE.        PTCN0616
C                                                                               PTCN0617
C   2.  CAREFULLY MONITOR THE VALUE OF IRET SO THAT ANY SERIOUS ERROR           PTCN0618
C   IS CAUGHT BEFORE ANOTHER CALL IS MADE TO PITCON.                            PTCN0619
C                                                                               PTCN0620
C   3.  CHOOSE A VALUE OF IMOD FOR THE TYPE OF CORRECTOR PROCESS TO             PTCN0621
C   BE USED.                                                                    PTCN0622
C                                                                               PTCN0623
C                                                                               PTCN0624
C   THE USER MAY -                                                              PTCN0625
C                                                                               PTCN0626
C   1.  MONITOR THE PASSING OF BIFURCATION POINTS BY SAVING THE OLD             PTCN0627
C   VALUE OF DETA AND COMPARING IT TO THE CURRENT VALUE.  IF THERE              PTCN0628
C   IS A CHANGE IN SIGN, THEN A BIFURCATION POINT HAS BEEN PASSED.              PTCN0629
C                                                                               PTCN0630
```

```
C  2.  ACCESS THE COMMON BLOCKS /COUNT1/ AND /COUNT2/ TO KEEP TRACK
C  OF THE AMOUNT OF WORK DONE.
C
C  3.  MONITOR THE COMMON BLOCK /POINT/ FOR INFORMATION ABOUT THE
C  SOLUTION CURVE.
C
C  4.  AT ANY TIME, RESET THE CODE BY PASSING IN KSTEP=-1 OR KSTEP=0.
C  THIS ALLOWS THE USER TO CHANGE STEPSIZE, DIRECTION OF CONTINUATION,
C  ERROR CONTROLS, OR OTHER PARAMETERS.  IT ALSO ENABLES THE USER
C  TO RUN UNRELATED PROBLEMS OF DIFFERENT SIZES OR ERROR CONTROLS
C  DURING A SINGLE PROGRAM EXECUTION.
C
C
C  THIS SUBROUTINE IS CALLED BY
C    USER MAIN PROGRAM
C  AND CALLS
C    CORECT
C    ROOT
C    TANGNT
C    FORTRAN ABS
C    FORTRAN ACOS
C    FORTRAN ALOG
C    FORTRAN AMAX1
C    FORTRAN AMIN1
C    FORTRAN DBLE
C    FORTRAN FLOAT
C    FORTRAN SIGN
C    FORTRAN SIN
C    FORTRAN SNGL
C    FORTRAN SQRT
C    LINPAK ISAMAX
C    LINPAK SAXPY
C    LINPAK SCOPY
C    LINPAK SNRM2
C    LINPAK SSCAL
C
C***********************************************************************
C
      INTEGER IPVT(NVAR)
      REAL RWORK(ISIZE)
      REAL WRGE(8),ACOF(12)
      DOUBLE PRECISION DTLIPC,DTCIPC,DADJUS,COSALF
      COMMON /COUNT1/ ICRSL,ITNSL,NSTCR,NCNCR,NTRCR,NLMCR,NLMRT
      COMMON /COUNT2/ IFEVAL,IPEVAL,ISOLVE,NRED,NRDSUM,KN,KNSUM
      COMMON /OUTPUT/ IWRITE
      COMMON /POINT/  DETA,IEXP,CURVCF,CORDIS,ALPHLC,HSECLC,FNRMXF
      COMMON /TOL/    EPMACH,EPSATE,EPSQRT
      DATA IDONE  /0/
      DATA TENM1  /0.1/
      DATA TENM2  /0.01/
      DATA TENM3  /0.001/
      DATA WRGE   /
     1 .8735115E+00, .1531947E+00, .3191815E-01, .3339946E-10,
     2 .4677788E+00, .6970123E-03, .1980863E-05, .1122789E-08/
      DATA ACOF   /
     1 .9043128E+00,-.7075675E+00,-.4667383E+01,-.3677482E+01,
     2 .8516099E+00,-.1953119E+00,-.4830636E+01,-.9770528E+00,
     3 .1040061E+01, .3793395E-01, .1042177E+01, .4450706E-01/
C
C***********************************************************************
C
C  1.  PREPARATIONS.
C  ON FIRST CALL FOR THIS PROBLEM, INITIALIZE COUNTERS AND VARIABLES,
C  CHECK USER INFORMATION AND SET DEFAULTS, AND IF (KSTEP.EQ.-1),
C  CHECK NORM OF F(XR) AND CORRECT XR IF NECESSARY.
C  ON EACH CALL, IF INPUT IRET HAS NONFATAL VALUE, RESET IRET
C  SO THAT CONTINUATION LOOP PICKS UP WHERE IT WAS HALTED.
C
C***********************************************************************
C
```

```
      IERR=0
      IF (IRET.EQ.-1) IRET=2
      IF (IRET.EQ.-2.OR.IRET.EQ.-3.OR.IRET.EQ.-4) IRET=1
      IF (IRET.EQ.-5.OR.IRET.EQ.-6) IRET=0
C
C  IF CODE WAS CALLED AGAIN AFTER FATAL VALUE OF IRET,
C  THEN RETURN WITH ERROR VALUE IRET=-10.
C
      IF (IRET.LT.0) GO TO 440
C
C  PERFORM ONE-TIME ONLY INITIALIZATIONS
C
      IF (IDONE.NE.0) GO TO 10
C
C  SET THE MACHINE DEPENDENT VARIABLE EPMACH, THE SMALLEST NUMBER
C  SO THAT (1.0+EPMACH.GT.1.0)
C
C  FOR DEC PDP-10 IN SINGLE PRECISION:
C
      EPMACH=7.4505806E-9
C
C  FOR IBM 360 OR 370 IN SHORT (SINGLE) PRECISION:
C
C     EPMACH=9.53674E-7
C
C  FOR CDC 6600 OR 7600 IN SINGLE PRECISION:
C
C     EPMACH=7.105427406E-15
C
C  SET EPSATE=8*EPMACH, EPSQRT=SQRT(EPMACH)
C
      EPSATE=8.0*EPMACH
      EPSQRT=SQRT(EPMACH)
      ALFMIN=2.0*ACOS(1.0-EPMACH)
      IF(KSTEP.LT.-1.OR.KSTEP.GT.0)KSTEP=-1
      KSTEPO=-2
      IDONE=1
C
C  PERFORM INITIALIZATIONS AND CHECKS FOR NEW PROBLEM ONLY
C
   10 IF (KSTEP.GT.0) GO TO 30
      IF(KSTEPO.EQ.-1.AND.KSTEP.EQ.0)GO TO 30
      IF (NVAR.LE.1) GO TO 440
      IF (ISIZE.LT.(NVAR)*(NVAR+5)) GO TO 440
      IXR=1
      JXR=0
      IXC=IXR+NVAR
      JXC=JXR+NVAR
      IXF=IXC+NVAR
      JXF=JXC+NVAR
      ITL=IXF+NVAR
      JTL=JXF+NVAR
      ITC=ITL+NVAR
      JTC=JTL+NVAR
      IFP=ITC+NVAR
      JFP=JTC+NVAR
      BETA=0.0
      TCIPC=0.0
      CORDIS=0.0
      CURVCF=0.0
      HSECLL=0.0
      HSECLC=0.0
      XITO=0.0
      ITO=0
      NEQN=NVAR-1
      NRED=0
      KNSUM=0
      NRDSUM=0
      ICRSL=0
      ITNSL=0
```

```
      NSTCR=0
      NCNCR=0
      NTRCR=0
      NLMCR=0
      NLMRT=0
      IFEVAL=0
      ISOLVE=0
      IPEVAL=0
      IF (HMAX.LE.0.0) HMAX=SQRT(FLOAT(NVAR))
      IF (HMIN.LE.EPSQRT) HMIN=EPSQRT
      HDEF=.5*(HMAX+HMIN)
      IF (HFACT.LE.1.0) HFACT=3.0
      HRED=1.0/HFACT
      IF (ABSERR.LE.0.0) ABSERR=EPSQRT
      IF (RELERR.LE.0.0) RELERR=EPSQRT
      IF (IPC.LE.0.OR.IPC.GT.NVAR) IPC=NVAR
      IF (LIM.LT.0.OR.LIM.GT.NVAR) LIM=0
      IF (H.EQ.0.0) H=HDEF
      DIR=SIGN(1.0,H)
      H=ABS(H)
C
C  IF (KSTEP.LT.0) CHECK NORM OF F(XR) AT STARTING POINT,
C  IF ACCEPTABLE, RETURN IMMEDIATELY WITH KSTEP=0,
C  OTHERWISE APPLY NEWTON'S METHOD, HOLDING VALUE OF
C  IPC-TH COMPONENT FIXED.
C
      IF (KSTEP.GE.0) GO TO 20
      CALL CORECT(NVAR,RWORK(IXR),IPC,RWORK(ITL),IERR,IMOD,RWORK(IFP),
     1 IPVT,ABSERR,RELERR,XSTEP,NEQN,FNRMXF)
      NSTCR=NSTCR+KN
C
C  IF NO ACCEPTABLE POINT FOUND, ERROR RETURN
C
      IF (IERR.NE.0) GO TO 400
      KSTEPO=-1
      KSTEP=0
      HTANCF=H
      GO TO 340
   20 IF (KSTEP.EQ.0) CALL SCOPY(NVAR,RWORK(IXR),1,RWORK(IXC),1)
      IF (KSTEP.EQ.0) CALL SCOPY(NVAR,RWORK(IXR),1,RWORK(IXF),1)
C
C***********************************************************************
C
C  2.  TARGET POINT CHECK.  IF (IT.NE.0) TARGET POINTS ARE SOUGHT.
C  CHECK TO SEE IF TARGET COMPONENT IT HAS VALUE XIT LYING
C  BETWEEN XC(IT) AND XF(IT).  IF SO, GET LINEARLY INTERPOLATED
C  STARTING POINT, AND USE NEWTON'S METHOD TO GET TARGET POINT
C
C***********************************************************************
C
   30 IF(IT.LT.0.OR.IT.GT.NVAR)IT=0
      IF (IT.EQ.0) GO TO 40
      IF (IRET.EQ.1.AND.XIT.EQ.XITO.AND.IT.EQ.ITO) GO TO 40
      XCIT=RWORK(JXC+IT)
      XFIT=RWORK(JXF+IT)
      IF ((XIT.LT.XCIT).AND.(XIT.LT.XFIT)) GO TO 40
      IF ((XIT.GT.XCIT).AND.(XIT.GT.XFIT)) GO TO 40
      DEL=XFIT-XCIT
      RAT=0.0
      IF (ABS(DEL).GT.EPSQRT) RAT=(XIT-XCIT)/DEL
      CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)
      CALL SAXPY(NVAR,-1.0,RWORK(IXC),1,RWORK(IXR),1)
      CALL SSCAL(NVAR,RAT,RWORK(IXR),1)
      CALL SAXPY(NVAR,1.0,RWORK(IXC),1,RWORK(IXR),1)
      RWORK(JXR+IT)=XIT
      CALL CORECT(NVAR,RWORK(IXR),IT,RWORK(ITL),IERR,IMOD,RWORK(IFP),
     1 IPVT,ABSERR,RELERR,XSTEP,NEQN,FNRM)
      NTRCR=NTRCR+KN
      ITO=IT
      XITO=XIT
```

```
      IF (IERR.EQ.0) GO TO 320
      IF (IERR.EQ.-1) GO TO 370
      IF (IERR.EQ.-2) GO TO 360
      IF (IERR.EQ.-3) GO TO 380
C
C***********************************************************************
C
C  3.  TANGENT AND LOCAL CONTINUATION PARAMETER CALCULATION.  UNLESS THE
C  TANGENT AND LIMIT POINT CALCULATIONS WERE ALREADY PERFORMED (BECAUSE
C  THE LOOP WAS INTERRUPTED FOR A LIMIT POINT), SET UP AND SOLVE
C  THE EQUATION FOR THE TANGENT VECTOR.  FORCE THE TANGENT VECTOR TO BE
C  OF UNIT LENGTH, AND FORCE THE IPL-COMPONENT TO HAVE THE SAME SIGN AS
C  THE IPL-TH COMPONENT OF THE PREVIOUS TANGENT VECTOR, OR (ON FIRST
C  STEP) THE SAME SIGN AS THE USER INPUT DIRECTION DIR.  SET THE LOCAL
C  PARAMETER IPC TO THE LOCATION OF THE LARGEST COMPONENT
C  OF THE TANGENT VECTOR, UNLESS A LIMIT POINT IN THAT DIRECTION
C  APPEARS TO BE APPROACHING AND ANOTHER CHOICE IS AVAILABLE.
C
C***********************************************************************
C
   40 IF (IRET.NE.2) GO TO 50
      IRET=0
      GO TO 160
C
C  STORE OLD TANGENT IN TL, COMPUTE NEW TANGENT FOR XC
C
   50 IPL=IPC
      IF(KSTEP.GT.0)CALL SCOPY(NVAR,RWORK(ITC),1,RWORK(ITL),1)
      ICALL=1
      CALL TANGNT(NVAR,RWORK(IXF),IPC,RWORK(ITC),IRET,ICALL,RWORK(IFP),
     1 IPVT,NEQN,DETA,IEXP)
      IF (IRET.EQ.-2) GO TO 430
      IF (IRET.EQ.-1) GO TO 410
C
C  SUBROUTINE TANGNT RETURNED IPC, THE LOCATION OF THE LARGEST COMPONENT
C  OF THE TANGENT VECTOR.  THIS WILL BE USED FOR THE LOCAL
C  PARAMETERIZATION OF THE CURVE UNLESS A LIMIT POINT IN IPC SEEMS
C  TO BE COMING.  TO CHECK THIS, WE COMPARE TCIPC:=TC(IPC) AND THE
C  SECOND LARGEST COMPONENT TCJPC:=TC(JPC).  IF TCJPC IS NO LESS
C  THAN .1 OF TCIPC, AND TC(JPC) IS LARGER THAN TL(JPC),
C  WHEREAS TC(IPC) IS LESS THAN TL(IPC), WE WILL RESET THE
C  LOCAL PARAMETER IPC:=JPC.
C
      TLIPL=TCIPC
      TCIPC=RWORK(JTC+IPC)
      JPC=IPC
      IF(ABS(TCIPC).GE.ABS(RWORK(JTL+IPC)))GO TO 60
      IF(TLIPL.EQ.0.0)GOTO 60
      RWORK(JTC+IPC)=0.0
      JPC=ISAMAX(NVAR,RWORK(ITC),1)
      TCJPC=RWORK(JTC+JPC)
      RWORK(JTC+IPC)=TCIPC
      IF (ABS(TCJPC).LT.TENM1*ABS(TCIPC)) GO TO 60
      IF (ABS(TCJPC).LT.ABS(RWORK(JTL+JPC)))GOTO 60
      IPC=JPC
      IF (IWRITE.GE.2)WRITE(6,610)
   60 TCIPL=RWORK(JTC+IPL)
      DTLIPC=DBLE(RWORK(JTL+IPC))
      DETA=DETA/TCIPL
C
C  ADJUST SIGN OF TANGENT
C  COMPARE THE SIGN OF TC(IPL) WITH SIGN OF TL(IPL)
C  (BUT ON FIRST STEP, COMPARE SIGN OF TC(IPL) WITH USER INPUT DIR).
C  IF THESE SIGNS DIFFER, CHANGE THE SIGN OF TC TO FORCE AGREEMENT,
C  AND THE SIGN OF DETA.
C  THEN RECORD DIR:= SIGN OF DETERMINANT = SIGN(DETA).
C
      STLIPL=DIR
      IF (TLIPL.NE.0.0) STLIPL=SIGN(1.0,TLIPL)
      IF (SIGN(1.0,TCIPL).EQ.STLIPL) GO TO 70
```

```
      CALL SSCAL(NVAR,-1.0,RWORK(ITC),1)
      DETA=-DETA
      TCIPL=-TCIPL
   70 TCIPC=RWORK(JTC+IPC)
      TCJPC=RWORK(JTC+JPC)
      DTCIPC=DBLE(TCIPC)
      DIR=SIGN(1.0,DETA)
      IF (LIM.EQ.0) GO TO 80
      TLLIM=TCLIM
      TCLIM=RWORK(JTC+LIM)
C
C  COMPUTE ALPHLC, THE ANGLE BETWEEN TANGENT AT XL AND TANGENT AT XC
C  AND HSECLC, THE EUCLIDEAN NORM OF SECANT FROM XL TO XC.
C
   80 IF(KSTEP.LE.0) GO TO 160
      COSALF=0.0D0
      DO 90 I=1,NVAR
           COSALF=COSALF+DBLE(RWORK(JTC+I))*DBLE(RWORK(JTL+I))
   90      RWORK(JXR+I)=RWORK(JXF+I)-RWORK(JXC+I)
      HSECLL=HSECLC
      HSECLC=SNRM2(NVAR,RWORK(IXR),1)
      ALPHLC=SNGL(COSALF)
      IF(ALPHLC.GT.1.0)ALPHLC=1.0
      IF(ALPHLC.LT.-1.0)ALPHLC=-1.0
      ALPHLC=ACOS(ALPHLC)
      IF(IWRITE.GE.2)WRITE(6,550)ALPHLC
C
C***********************************************************************
C
C  4.  LIMIT POINT CHECK.  IF (LIM.NE.0) CHECK TO SEE IF
C  OLD AND NEW TANGENTS DIFFER IN SIGN OF LIM-TH COMPONENT.
C  IF SO, ATTEMPT TO COMPUTE A POINT XR BETWEEN XC AND XF
C  FOR WHICH TANGENT COMPONENT VANISHES
C
C***********************************************************************
C
      IF (LIM.LE.0.OR.KSTEP.LE.0) GO TO 160
C
C  CHECK FOR LIMIT INTERVAL
C
      IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160
C
C  TEST FOR ACCEPTABLE ENDPOINTS
C
      ATLLM=ABS(TLLIM)
      IF (ATLLM.GT.0.5*ABSERR) GO TO 110
C
C  IF XC IS LIMIT POINT, TL ALREADY CONTAINS TANGENT AT XC
C
  100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1)
      GO TO 310
  110 ATCLIM=ABS(TCLIM)
      IF (ATCLIM.GT.0.5*ABSERR) GO TO 130
  120 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)
      CALL SCOPY(NVAR,RWORK(ITC),1,RWORK(ITL),1)
      GO TO 310
C
C  TEST FOR SMALL INTERVAL
C
  130 XCLIM=RWORK(JXC+LIM)
      XFLIM=RWORK(JXF+LIM)
      DEL=ABS(XFLIM-XCLIM)
      XABS=AMAX1(ABS(XCLIM),ABS(XFLIM))
      IF (DEL.GT.(ABSERR+RELERR*XABS)) GO TO 140
      IF (ATLLM.GT.ATCLIM) GO TO 120
      GO TO 100
C
C  BEGIN ROOT-FINDING ITERATION WITH INTERVAL (0,1) AND
C  FUNCTION VALUES TL(LIM), TC(LIM).
C
```

```
  140 KOUNT=0
      A=0.0
      FA=TLLIM
      TSN=TCLIM
      C=1.0
      FC=TCLIM
C
C  SET IPLIM TO THE INDEX OF MAXIMUM ENTRY OF SECANT
C  (EXCEPT THAT IPLIM MUST NOT EQUAL LIM)
C  AND SAVE THE SIGN OF THE MAXIMUM COMPONENT IN DIRLIM
C  SO THAT NEW TANGENTS MAY BE PROPERLY SIGNED.
C
      CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)
      CALL SAXPY(NVAR,-1.0,RWORK(IXC),1,RWORK(IXR),1)
      RWORK(JXR+LIM)=0.0
      IPLIM=ISAMAX(NVAR,RWORK(IXR),1)
      DIRLIM=SIGN(1.0,RWORK(JXR+IPLIM))
C
C  CALL ROOTFINDER FOR APPROXIMATE ROOT SN, SET X=SN*XF+(1-SN)*XC
C  CALL CORRECTOR TO RETURN TO CURVE ON LINE X(IPLIM)=CONSTANT,
C  COMPUTE TANGENT THERE, AND SET FUNCTION VALUE TO TANGENT(LIM)
C
  150 CALL ROOT(A,FA,SN,TSN,C,FC,KOUNT,IFLAG)
      NLMRT=NLMRT+1
      IF(IFLAG.LT.-1)GO TO 350
      IF(IFLAG.EQ.-1.OR.IFLAG.EQ.0)GO TO 310
      CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)
      CALL SSCAL(NVAR,SN,RWORK(IXR),1)
      SCALER=1.0-SN
      CALL SAXPY(NVAR,SCALER,RWORK(IXC),1,RWORK(IXR),1)
      CALL CORECT(NVAR,RWORK(IXR),IPLIM,RWORK(ITL),IERR,IMOD,RWORK(IFP)
     1 ,IPVT,ABSERR,RELERR,XSTPLM,NEQN,FNRM)
      NLMCR=NLMCR+KN
      IF (IERR.NE.0) GO TO 350
      ICALL=1
      IPT=IPLIM
      CALL TANGNT(NVAR,RWORK(IXR),IPT,RWORK(ITL),IRET,ICALL,
     1 RWORK(IFP),IPVT,NEQN,DETLIM,IEXLIM)
      IF(IRET.LT.0)GO TO 350
C
C  ADJUST THE SIGN OF THE TANGENT SO THAT THE IPLIM-TH COMPONENT
C  HAS THE SAME SIGN AS THE IPLIM-TH COMPONENT OF THE SECANT
C
      IF (DIRLIM.NE.SIGN(1.0,RWORK(JTL+IPLIM)))
     1 CALL SSCAL(NVAR,-1.0,RWORK(ITL),1)
C
C  SEE IF WE CAN ACCEPT THE NEW POINT BECAUSE TANGENT(LIM) IS SMALL
C  OR MUST 'ACCEPT' THE POINT BECAUSE THE VALUES ARE NOT DECREASING
C  RAPIDLY ENOUGH, OR IF WE CAN GO ON.
C
      TSHOLD=TSN
      TSN=RWORK(JTL+LIM)
      IF (ABS(TSN).LE.0.5*ABSERR) GO TO 310
      GO TO 150
C
C***********************************************************************
C
C  5.  STEP LENGTH COMPUTATION.  COMPUTE STEPLENGTH HTANCF
C
C  THE FORMULAS UNDERLYING THE ALGORITHM ARE
C
C  LET
C
C  ALPHLC = MAXIMUM OF ARCCOS(TL,TC) AND ALFMIN = 2*ARCCOS(1-EPMACH)
C  HSECLC = NORM(XL-XC)
C  HSECLL = NORM(XL-XLL)
C  ABSIN  = ABS(SIN(.5*ALPHLC))
C  CURVLC = LAST VALUE OF CURVCF
C  CURVCF = 2*ABSIN/HSECLC
C  CORDIS = OPTIMIZED CORRECTOR DISTANCE TO CURRENT CONTINUATION POINT.
```

```
C             BUT CORDIS FORCED TO LIE BETWEEN .1*HSECLC AND HSECLC.
C             UNLESS (CORDIS.EQ.0.0), BECAUSE THE PREDICTED POINT WAS
C             IMMEDIATELY ACCEPTED.  IN SUCH A CASE, SET HTANCF=HSECLC
C             INSTEAD OF USING FIRST ESTIMATE FOR HTANCF.
C
C   THEN
C
C   CURVXF = CURVCF + HSECLC*(CURVCF-CURVLC)/(HSECLC+HSECLL)
C   BUT CURVXF MUST BE GREATER THAN .001, AND A SIMPLER FORMULA IS USED
C   IF WE DO NOT HAVE DATA AT TWO PREVIOUS POINTS.
C
C   FIRST ESTIMATE:  (UNLESS (CORDIS.EQ.0.0) )
C
C   HTANCF  = SQRT(2*CORDIS/CURVXF)
C
C   ADJUSTED VALUE:
C
C   HTANCF = HTANCF*(1.0 + HTANCF*(TC(IPC)-TL(IPC))/(2*HSECLC*TC(IPC)))
C
C   READJUSTMENT AND TRUNCATIONS:
C
C   IF STEPSIZE REDUCTION OCCURRED DURING LAST CORRECTOR PROCESS,
C   HTANCF IS FORCED TO BE LESS THAN (HFACT-1)*HSECLC/2.
C
C   HTANCF MUST LIE BETWEEN (HSECLC/HFACT) AND (HSECLC*HFACT).
C
C   HTANCF IS ALWAYS FORCED TO LIE BETWEEN HMIN AND HMAX.
C
C***********************************************************************
C
C   CHECK IF DEFAULT STEP MUST BE USED:
C   ON FIRST STEP, USE HTANCF=H.
C   IF PREVIOUS STEP WAS OF SIZE ZERO, USE STEPSIZE HDEF=(HMIN+HMAX)/2
C
  160 IF (KSTEP.GT.0.AND.HSECLC.GT.0.0) GO TO 170
      HTANCF=HDEF
      IF(KSTEP.LE.0)HTANCF=H
      GO TO 190
  170 IF(ALPHLC.LT.ALFMIN)ALPHLC=ALFMIN
      ABSIN=ABS(SIN(.5*ALPHLC))
C
C   COMPUTE NEW CURVATURE DATA
C
      CURVLC=CURVCF
      CURVCF=2.0*ABSIN/HSECLC
      CURVXF=CURVCF
      IF(HSECLL.NE.0.0)
     1 CURVXF=CURVCF+HSECLC*(CURVCF-CURVLC)/(HSECLC+HSECLL)
      CURVXF=AMAX1(CURVXF,TENM3)
      IF(IWRITE.GE.2)WRITE(6,560)CURVCF,CURVXF
C
C   IF THE CONVERGENCE DISTANCE IS ZERO, SET FIRST ESTIMATE TO HSECLC.
C   OTHERWISE, TRUNCATE CORDIS TO LIE BETWEEN .01*HSECLC AND HSECLC.
C
      HTANCF=HSECLC
      IF(CORDIS.EQ.0.0) GO TO 180
      TEMP=TENM2*HSECLC
      CORDIS=AMAX1(CORDIS,TEMP)
      CORDIS=AMIN1(CORDIS,HSECLC)
C
C   SET HTANCF, THEN ADJUST FOR CURVATURE IN CONTINUATION
C   PARAMETER DIRECTION, THEN TRUNCATE
C
      HTANCF=SQRT(2.0*CORDIS/CURVXF)
  180 IF(NRED.GT.0)HTANCF=AMIN1(HTANCF,(HFACT-1.0)*HSECLC*.5)
      DADJUS=1.0D0+(1.0D0-DTLIPC/DTCIPC)*DBLE(.5*HTANCF)/DBLE(HSECLC)
      HTANCF=HTANCF*SNGL(DADJUS)
      TEMP=HSECLC*HRED
      HTANCF=AMAX1(HTANCF,TEMP)
      TEMP=HSECLC*HFACT
```

```
      HTANCF=AMIN1(HTANCF,TEMP)
      HTANCF=AMAX1(HTANCF,HMIN)
      HTANCF=AMIN1(HTANCF,HMAX)
C
C***********************************************************************
C
C  6.  PREDICTION AND CORRECTION STEPS.  USING XR=XC+HTANCF*TC
C  AS STARTING POINT, CORRECT XR WITH A FULL OR MODIFIED NEWTON
C  ITERATION.  IF CORECT FAILS, REDUCE STEPSIZE USED FOR PREDICTOR
C  POINT, AND TRY AGAIN.  CORRECTION WILL ONLY BE ABANDONED IF STEPSIZE
C  FALLS BELOW HMIN.
C
C***********************************************************************
C
  190 KSTEPO=KSTEP
      KSTEP=KSTEP+1
      NRED=0
  200 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)
      CALL SAXPY(NVAR,HTANCF,RWORK(ITC),1,RWORK(IXR),1)
  210 IF(IWRITE.GE.2)WRITE(6,570)HTANCF
      IF(IWRITE.GE.3)WRITE(6,580)(RWORK(IXR+I),I=1,NVAR)
      CALL CORECT(NVAR,RWORK(IXR),IPC,RWORK(ITL),IERR,IMOD,RWORK(IFP),
     1 IPVT,ABSERR,RELERR,XSTEP,NEQN,FNRMXF)
      NCNCR=NCNCR+KN
      IF (IERR.EQ.0) GO TO 230
      IF (IERR.EQ.-1) GO TO 420
C
C  NO CONVERGENCE, TRY A SMALLER STEPSIZE
C
      HTANCF=HRED*HTANCF
      IF (HTANCF.LT.HMIN) GO TO 390
      NRED=NRED+1
      IF (IERR.EQ.-2) GO TO 220
      GO TO 200
  220 CALL SAXPY(NVAR,-1.0,RWORK(IXC),1,RWORK(IXR),1)
      CALL SSCAL(NVAR,HRED,RWORK(IXR),1)
      CALL SAXPY(NVAR,1.0,RWORK(IXC),1,RWORK(IXR),1)
      GO TO 210
C
C***********************************************************************
C
C  7.  SUCCESSFUL STEP.  STORE INFORMATION BEFORE RETURN.
C  UPDATE OLD AND CURRENT CONTINUATION POINTS.
C  COMPUTE CORDIS, THE SIZE OF THE CORRECTOR STEP.  COMPUTE
C  A FACTOR THETA WHICH RESCALES CORDIS TO A VALUE WHICH WOULD
C  CORRESPOND TO A DESIRABLE NUMBER OF CORRECTOR STEPS
C  (4 FOR FULL NEWTON, 10 FOR MODIFIED NEWTON).
C  SEE REFERENCE DEN HEIJER AND RHEINBOLDT, LOC. CIT.
C
C***********************************************************************
C
  230 NRDSUM=NRDSUM+NRED
      IF(NRED.NE.0.AND.IWRITE.GE.2)WRITE(6,590)NRED
C
C  COMPUTE CORRECTOR STEP = XC+HTANCF*TC-XF
C  SET CORDIS = MAX NORM OF CORRECTOR STEP
C
      CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXC),1)
      CALL SAXPY(NVAR,-1.0,RWORK(IXR),1,RWORK(IXC),1)
      CALL SAXPY(NVAR,HTANCF,RWORK(ITC),1,RWORK(IXC),1)
      IMAX=ISAMAX(NVAR,RWORK(IXC),1)
      CORDIS=ABS(RWORK(IXC+IMAX))
      IF(KN.EQ.0) CORDIS=0.0
C
C  MODIFY CORDIS TO A VALUE THAT WOULD CORRESPOND TO THE
C  DESIRED NUMBER OF CORRECTOR STEPS
C
      IF(CORDIS.EQ.0.0)GO TO 300
      OMEGA=XSTEP/CORDIS
      THETA=0.0
```

```
      IF(IMOD.EQ.1) GOTO 260
C
C  FULL NEWTON METHOD FOR CORRECTOR STEPS
C
      IF(KN.LE.1)THETA=8.0
      IF(KN.EQ.4)THETA=1.0
      IF(THETA.NE.0.0)GO TO 290
      IF(KN.GT.4)GO TO 240
      LK=4*KN-7
      THETA=1.0
      IF(OMEGA.GE.WRGE(LK))GOTO 290
      LST=LK
      IF(OMEGA.GE.WRGE(LK+1))GOTO 250
      LST=LK+2
      IF(OMEGA.GE.WRGE(LK+2))GOTO 250
      THETA=8.0
      GOTO 290
  240 THETA=0.125
      IF(KN.GE.7) GOTO 290
      LK=4*KN-16
      IF(OMEGA.LE.WRGE(LK))GOTO 290
      LST=2*KN-1
  250 THETA=ACOF(LST)+ACOF(LST+1)*ALOG(OMEGA)
      GOTO 290
C
C  MODIFIED NEWTON METHOD FOR CORRECTOR STEPS
C
  260 IF(KN.LE.1)THETA=8.0
      IF(KN.EQ.10)THETA=1.0
      IF(THETA.NE.0.0)GO TO 290
      EXPON=FLOAT(KN-1)/FLOAT(KN-10)
C
C  AVOID OVERFLOW OR UNDERFLOW BY ANTICIPATING
C  CUTOFF VALUES OF THETA
C
      IF (KN.GT.10) GO TO 270
      IF (8.0**EXPON.GT.OMEGA) THETA=8.0
      IF (.125**EXPON.LT.OMEGA) THETA=.125
      IF (THETA.NE.0.0) GO TO 290
      GO TO 280
  270 IF (8.0**EXPON.LT.OMEGA) THETA=8.0
      IF (.125**EXPON.GT.OMEGA) THETA=.125
      IF (THETA.NE.0.0) GO TO 290
  280 EXPON=1.0/EXPON
      THETA=OMEGA**EXPON
      THETA=AMAX1(THETA,0.125)
      THETA=AMIN1(THETA,8.0)
C
C  SET THE MODIFIED VALUE OF CORDIS
C
  290 CORDIS=THETA*CORDIS
      IF(IWRITE.GE.2)WRITE(6,600)OMEGA,THETA,CORDIS
  300 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXC),1)
      CALL SCOPY(NVAR,RWORK(IXR),1,RWORK(IXF),1)
      GO TO 340
C
C***********************************************************************
C
C  RETURNS.  SET VALUE OF IRET.  IF AN ERROR OCCURRED, PRINT
C  A MESSAGE AS WELL.
C
C***********************************************************************
C
C
C  RETURN LIMIT POINT
C
  310 IRET=2
      RETURN
C
C  RETURN WITH TARGET POINT
```

```
C
  320 IRET=1
      RETURN
C
C  RETURN WITH CONTINUATION POINT
C
  330 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)
  340 IRET=0
      H=HTANCF
      RETURN
C
C  ERROR RETURNS
C
  350 IRET=-1
      IF(IWRITE.GE.1)WRITE(6,450)
      RETURN
  360 IRET=-2
      IF(IWRITE.GE.1)WRITE(6,460)
      RETURN
  370 IRET=-3
      IF(IWRITE.GE.1)WRITE(6,470)
      RETURN
  380 IRET=-4
      IF(IWRITE.GE.1)WRITE(6,480)
      RETURN
  390 IRET=-5
      IF(IWRITE.GE.1)WRITE(6,490)HTANCF,HMIN
      RETURN
  400 IRET=-6
      IF(IWRITE.GE.1)WRITE(6,500)
      RETURN
  410 IRET=-7
      IF(IWRITE.GE.1)WRITE(6,510)
      RETURN
  420 IRET=-8
      IF(IWRITE.GE.1)WRITE(6,520)
      RETURN
  430 IRET=-9
      IF(IWRITE.GE.1)WRITE(6,530)
      RETURN
  440 IRET=-10
      IF(IWRITE.GE.1)WRITE(6,540)NVAR,ISIZE
      RETURN
  450 FORMAT(26HOLIMIT POINT FINDER FAILED)
  460 FORMAT(50HOCORECT, SEEKING TARGET POINT, TOOK TOO MANY STEPS)
  470 FORMAT(50HOCORECT, SEEKING TARGET, CALLED SOLVE WHICH FAILED)
  480 FORMAT(45HOCORECT, SEEKING TARGET, FAILED WITH BAD STEP)
  490 FORMAT(10HOSTEPSIZE ,F12.7,15H LESS THAN HMIN,F12.7)
  500 FORMAT(42HONORM OF F(X) IS TOO LARGE ON INITIAL CALL)
  510 FORMAT(33HOSOLVE FAILED IN CALL FROM TANGNT)
  520 FORMAT(33HOSOLVE FAILED IN CALL FROM CORECT)
  530 FORMAT(23HOTANGENT VECTOR IS ZERO)
  540 FORMAT(26HOUNACCEPTABLE INPUT  NVAR=,I10,7H ISIZE=,I10)
  550 FORMAT(36H ANGLE BETWEEN OLD AND NEW TANGENTS=,F12.5)
  560 FORMAT(9H CURVCF =,E14.6,9H CURVXF =,E14.6)
  570 FORMAT(16H USING STEPSIZE=,F12.5)
  580 FORMAT(12H PREDICTED X/1X,5F12.5)
  590 FORMAT(1H ,I2,16H STEP REDUCTIONS)
  600 FORMAT(7H OMEGA=,F12.5,7H THETA=,F12.5,9H NEW RAD=,F12.5)
  610 FORMAT(31H TANGNT ANTICIPATES LIMIT POINT)
C
C***********************************************************************
C
      END
```

```
      SUBROUTINE CORECT(NVAR,X,IHOLD,WORK,IERR,IMOD,FPRYM,IPVT,
     1 ABSERR,RELERR,XSTEP,NEQN,FNRM)
C
C***********************************************************************
C
C  SUBROUTINE CORECT PERFORMS THE CORRECTOR ITERATIONS ON A STARTING
C  POINT.  THE CORRECTION METHOD IS EITHER FULL (IMOD=0)
C  OR MODIFIED (IMOD=1) NEWTON'S METHOD.  FOR MODIFIED NEWTON'S
C  METHOD, THE JACOBIAN IS TO BE EVALUATED ONLY AT THE STARTING POINT.
C  IF B IS THE VALUE OF X(IHOLD) FOR THE INPUT STARTING POINT,
C  THEN THE AUGMENTING EQUATION IS X(IHOLD)=B,
C  THAT IS, THE IHOLD-TH COMPONENT OF X IS TO BE HELD FIXED.
C  THE AUGMENTED SYSTEM TO BE SOLVED IS THEN DFA(X,IHOLD)*DELTA=FA(X)
C
C  INPUT
C  X     = THE STARTING POINT FOR THE CORRECTOR ITERATION.
C  IHOLD = COMPONENT OF X THAT WILL NOT BE CHANGED DURING ITERATION
C  IMOD  = FLAG FOR TYPE OF NEWTON'S METHOD TO BE USED.
C          WHEN IMOD=0, JACOBIAN IS TO BE EVALUATED AT EVERY
C          CORRECTOR ITERATE.  KNMAX IS SET TO 10
C          IF IMOD=1, THE JACOBIAN IS ONLY EVALUATED AT THE STARTING
C          POINT, AND KNMAX IS SET TO 20.
C
C  OUTPUT
C  X     = SOLUTION VECTOR ON A SUCCESSFUL CALL TO CORECT.
C  WORK  = THE RESIDUAL F(X), AFTER A SUCCESSFUL CALL TO CORECT.
C  IERR  = THE RETURN FLAG WITH THE FOLLOWING VALUES
C         -2 MAXIMUM NUMBER OF CORRECTOR ITERATIONS WERE TAKEN.
C         -1 ERROR RETURN FROM SOLVE CALLED BY CORECT.
C          0 SUCCESSFUL CORRECTION.  VECTOR X RETURNED SATISIFES
C            ABS(F(X)).LE.ABSERR
C          1 CORRECTOR STEP WAS UNACCEPTABLE, CORRECTION FAILED.
C  KN    = THE NUMBER OF CORRECTOR ITERATIONS TAKEN ON THIS CALL
C
C  THIS SUBROUTINE IS CALLED BY
C    PITCON
C  AND CALLS
C    SOLVE
C    FORTRAN ABS
C    LINPAK ISAMAX
C    LINPAK SAXPY
C    USER FCTN
C
C***********************************************************************
C
      REAL X(NVAR),WORK(NVAR),FPRYM(NVAR,NVAR)
      INTEGER IPVT(NVAR)
      COMMON /COUNT1/ ICRSL,ITNSL,NSTCR,NCNCR,NTRCR,NLMCR,NLMRT
      COMMON /COUNT2/ IFEVAL,IPEVAL,ISOLVE,NRED,NRDSUM,KN,KNSUM
      COMMON /OUTPUT/ IWRITE
      COMMON /TOL/    EPMACH,EPSATE,EPSQRT
C
C  INITIALIZE
C
      KN=0
      KNMAX=10
      IF(IMOD.EQ.1)KNMAX=20
      IERR=0
      FMP=2.0
      ICALL=1
      XSTEP=0.0
      CALL FCTN(NVAR,X,WORK)
      IFEVAL=IFEVAL+1
      IMAX=ISAMAX(NEQN,WORK,1)
      FNRM=ABS(WORK(IMAX))
      WORK(NVAR)=0.0
C
C  STRICTER ABSERR TEST ON STARTING POINT
C
      IF (FNRM.LE.0.5*ABSERR) GO TO 60
```

```
C
C  ITERATION LOOP
C
      DO 20 I=1, KNMAX
         KN=I
         CALL SOLVE(NVAR,X,WORK,IHOLD,DETA,IEXP,IERR,ICALL,IMOD,FPRYM,
     1   IPVT)
         ICRSL=ICRSL+1
         IF(IMOD.EQ.1)ICALL=0
         ISOLVE=ISOLVE+1
         IF (IERR.NE.0) GO TO 50
         FNRML=FNRM
         XSTEPL=XSTEP
         CALL SAXPY(NVAR,-1.0,WORK,1,X,1)
         IMAX=ISAMAX(NVAR,WORK,1)
         XSTEP=ABS(WORK(IMAX))
         IMAX=ISAMAX(NVAR,X,1)
         XNRM=ABS(X(IMAX))
         CALL FCTN(NVAR,X,WORK)
         IFEVAL=IFEVAL+1
         IMAX=ISAMAX(NEQN,WORK,1)
         FNRM=ABS(WORK(IMAX))
         WORK(NVAR)=0.0
C
C  ACCEPTANCE TEST
C
         IF (FNRM.LE.EPSATE) GO TO 60
         IF (FNRM.GT.ABSERR) GO TO 10
         IF (XSTEP.LE.(ABSERR+RELERR*XNRM)) GO TO 60
C
C  REJECTION TEST
C
   10    IF (KN.GT.1.AND.XSTEP.GT.(FMP*XSTEPL)) GO TO 30
         IF (FNRM.GT.(FMP*FNRML)) GO TO 30
   20    FMP=1.05
      GO TO 40
C
C  UNSUCCESSFUL STEP
C
   30 IERR=-3
      IF(IWRITE.EQ.2)WRITE(6,120)
      GO TO 70
C
C  MAXIMUM NUMBER OF CORRECTOR STEPS REACHED
C
   40 IERR=-2
      IF(IWRITE.EQ.2)WRITE(6,110)
      GO TO 70
C
C  ERROR RETURN IN SOLVE
C
   50 IERR=-1
      IF(IWRITE.EQ.2)WRITE(6,100)
      GO TO 70
C
C  SUCCESSFUL STEP
C
   60 IERR=0
   70 KNSUM=KNSUM+KN
      IF(IWRITE.EQ.2)WRITE(6,80) KN,XSTEP
      IF(IWRITE.EQ.2)WRITE(6,90)IHOLD
      RETURN
   80 FORMAT(13H CORECT TOOK ,I2,21H STEPS, LAST ONE WAS ,E12.5)
   90 FORMAT(14H CORECT IHOLD=,I3)
  100 FORMAT(31H0SOLVE FAILED, CALLED BY CORECT)
  110 FORMAT(25H0TOO MANY CORRECTOR STEPS)
  120 FORMAT(24H0CORRECTOR STEP REJECTED)
C
C***********************************************************************
C
      END
```

```
      SUBROUTINE TANGNT(NVAR,X,IP,TAN,IRET,ICALL,FPRYM,IPVT,NEQN,DETA,
     1 IEXP)
C
C***********************************************************************
C
C  SUBROUTINE TANGNT COMPUTES THE UNIT TANGENT VECTOR TO THE SOLUTION
C  CURVE OF THE UNDERDETERMINED NONLINEAR SYSTEM FX = 0.  THE
C  TANGENT VECTOR TAN IS THE SOLUTION OF THE LINEAR SYSTEM
C
C        DFA(X,IPL)*TAN = E(NVAR)
C
C  WHERE DFA(X,IPL) IS THE NVAR BY NVAR MATRIX WHOSE FIRST  NVAR-1  ROWS
C  ARE DFX/DX (X), THE DERIVATIVE OF FX EVALUATED AT X, AND WHOSE LAST
C  ROW IS (E(IPL)) TRANSPOSE, THE NVAR COMPONENT EUCLIDEAN COORDINATE
C  VECTOR WITH 1 IN THE IPL-TH POSITION AND ZEROS ELSEWHERE.  E(NVAR) IS
C  THE NVAR COMPONENT EUCLIDEAN COORDINATE VECTOR WITH ONE IN THE
C  LAST COMPONENT.
C    THE TANGENT VECTOR IS THEN NORMALIZED AND ITS SIGN ADJUSTED.
C
C  INPUT
C  NVAR  = THE NUMBER OF VARIABLES
C  X     = THE CURRENT CONTINUATION POINT
C  IP    = CONTINUATION COMPONENT SET ON LAST STEP
C
C  OUTPUT
C  TAN   = THE UNIT TANGENT VECTOR IN CONTINUATION DIRECTION AT X
C  DETA  = BINARY MANTISSA OF DETERMINANT OF JACOBIAN DFA(X,IPL)
C  IEXP  = BINARY EXPONENT OF THE DETERMINANT OF JACOBIAN DFA(X,IPL)
C  IP    = LOCATION OF LARGEST COMPONENT OF TANGENT VECTOR TAN
C          CANDIDATE FOR NEW CONTINUATION COMPONENT
C
C  THIS SUBROUTINE IS CALLED BY
C    PITCON
C  AND CALLS
C    SOLVE
C    LINPAK ISAMAX
C    LINPAK SNRM2
C    LINPAK SSCAL
C
C***********************************************************************
C
      REAL X(NVAR),TAN(NVAR),FPRYM(NVAR,NVAR)
      INTEGER IPVT(NVAR)
      COMMON /COUNT1/ ICRSL,ITNSL,NSTCR,NCMCR,NTRCR,NLMCR,NLMRT
      COMMON /COUNT2/ IFEVAL,IPEVAL,ISOLVE,NRED,NRDSUM,KN,KNSUM
      COMMON /OUTPUT/ IWRITE
C
C  COMPUTE TANGENT VECTOR
C
      DO 10 I=1,NEQN
   10      TAN(I)=0.0
      TAN(NVAR)=1.0
      IERR=0
      CALL SOLVE(NVAR,X,TAN,IP,DETA,IEXP,IERR,ICALL,IMOD,FPRYM,
     1 IPVT)
      ITNSL=ITNSL+1
      ISOLVE=ISOLVE+1
      IF (IERR.NE.0) IRET=-1
      IF (IRET.LT.0) RETURN
C
C  OBTAIN EUCLIDEAN NORM OF TANGENT VECTOR
C
      IP=ISAMAX(NVAR,TAN,1)
      TNORM=SNRM2(NVAR,TAN,1)
      IF (TNORM.EQ.0.0) IRET=-2
      IF (IRET.LT.0) RETURN
C
C  NORMALIZE THE VECTOR
C
      SCALER=1.0/TNORM
      CALL SSCAL(NVAR,SCALER,TAN,1)
      RETURN
C
C***********************************************************************
C
      END
```

```
      SUBROUTINE ROOT(A,FA,B,FB,C,FC,KOUNT,IFLAG)
C
C***********************************************************************
C
C  SUBROUTINE ROOT SEEKS A ROOT OF THE EQUATION F(X)=0.0,
C  GIVEN A STARTING INTERVAL (A,C) ON WHICH F CHANGES SIGN.
C  ON FIRST CALL TO ROOT, THE INTERVAL AND FUNCTION VALUES
C  FA AND FC ARE FED IN AND AN APPROXIMATION B FOR THE ROOT IS RETURNED.
C  BEFORE EACH SUBSEQUENT CALL, THE USER EVALUATES FB=F(B), AND THE
C  PROGRAM TRIES TO RETURN A BETTER APPROXIMATION B.
C
C  THIS PROGRAM IS BASED ON THE FORTRAN FUNCTION ZERO
C  GIVEN IN THE BOOK:
C  'ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES'
C  BY RICHARD P. BRENT, PRENTICE HALL,INC, 1973
C
C  THE MODIFICATIONS WERE DONE BY JOHN BURKARDT.
C
C  ON INPUT:
C
C  A      - IS ONE ENDPOINT OF AN INTERVAL IN WHICH F CHANGES SIGN.
C  FA     - THE VALUE OF F(A).  THE USER MUST EVALUATE F(A) BEFORE FIRST
C           CALL ONLY.  THEREAFTER THE PROGRAM SETS FA.
C  B      - ON FIRST CALL, B SHOULD NOT BE SET BY THE USER.
C           ON SUBSEQUENT CALLS, B SHOULD NOT BE CHANGED
C           FROM ITS OUTPUT VALUE, THE CURRENT APPROXIMANT
C           TO THE ROOT.
C  FB     - ON FIRST CALL, FB SHOULD NOT BE SET BY THE USER.
C           THEREAFTER, THE USER SHOULD EVALUATE THE FUNCTION
C           AT THE OUTPUT VALUE B, AND RETURN FB=F(B).
C  C      - IS THE OTHER ENDPOINT OF THE INTERVAL IN WHICH
C           F CHANGES SIGN.  NOTE THAT THE PROGRAM WILL RETURN
C           IMMEDIATELY WITH AN ERROR FLAG IF FC*FA.GT.0.0.
C  FC     - THE VALUE OF F(C).  THE USER MUST EVALUATE F(C) BEFORE FIRST
C           CALL ONLY.  THERAFTER THE PROGRAM SETS FC.
C  KOUNT  - A COUNTER FOR THE NUMBER OF CALLS TO ROOT.  KOUNT
C           SHOULD BE SET TO ZERO ON THE FIRST CALL FOR A GIVEN
C           ROOT PROBLEM.
C  IFLAG  - AN ERROR RETURN FLAG WHOSE INPUT VALUE IS IMMATERIAL.
C
C  ON RETURN FROM A CALL TO ROOT
C
C  A      - ONE ENDPOINT OF CHANGE OF SIGN INTERVAL.
C  FA     - THE VALUE OF F(A).
C  B      - CURRENT APPROXIMATION TO THE ROOT.  BEFORE ANOTHER
C           CALL TO ROOT, EVALUATE F(B).
C  FB     - FB WILL BE OVERWRITTEN BY THE USER BEFORE ANOTHER
C           CALL.  ITS VALUE ON RETURN IS ONE OF FA, FB OR FC.
C  C      - OTHER ENDPOINT OF CHANGE IN SIGN INTERVAL.
C  FC     - THE VALUE OF F(C).
C  KOUNT  - CURRENT NUMBER OF CALLS TO ROOT.
C  IFLAG  - PROGRAM RETURN FLAG:
C           IFLAG=-2 MEANS THAT ON FIRST CALL, FA*FC.GT.0.0.
C                    THIS IS AN ERROR RETURN, SINCE A BRACKETING
C                    INTERVAL SHOULD BE SUPPLIED ON FIRST CALL.
C           IFLAG=-1 MEANS THAT THE CURRENT BRACKETING INTERVAL
C                    WHOSE ENDPOINTS ARE STORED IN A AND C
C                    IS SO SMALL (LESS THAN 4*EPMACH*ABS(B)+EPMACH)
C                    THAT B SHOULD BE ACCEPTED AS THE ROOT.
C                    THE FUNCTION VALUE F(B) IS STORED IN FB.
C           IFLAG= 0 MEANS THAT THE INPUT VALUE FB IS EXACTLY
C                    ZERO, AND B SHOULD BE ACCEPTED AS THE ROOT.
C
C           IFLAG.GT.0  MEANS THAT THE CURRENT APPROXIMATION TO
C                    THE ROOT IS CONTAINED IN B.  IF A BETTER
C                    APPROXIMATION IS DESIRED, SET FB=F(B)
C                    AND CALL ROOT AGAIN.  THE VALUE OF IFLAG INDICATES
C                    THE METHOD THAT WAS USED TO PRODUCE B.
C
C           IFLAG= 1 BISECTION WAS USED.
```

```
C            IFLAG= 2 LINEAR INTERPOLATION (SECANT METHOD).
C            IFLAG= 3 INVERSE QUADRATIC INTERPOLATION.
C
C  LOCAL VARIABLES INCLUDE:
C
C  EPMACH- SMALLEST POSITIVE NUMBER SUCH THAT 1.0+EPMACH.GT.1.0
C          .5*BETA**(1-TAU) FOR ROUNDED, TAU-DIGIT ARITHMETIC
C          BASE BETA.  TWICE THAT VALUE FOR TRUNCATED ARITHMETIC.
C          THIS IS THE RELATIVE MACHINE PRECISION.
C  HALFBC- SIGNED HALFWIDTH OF INTERVAL.  DURING SEGMENT 3, THE
C          CHANGE OF SIGN INTERVAL IS (B,C) OR (C,B).  THE MIDPOINT
C          OF THAT INTERVAL IS XMID=B+HALFBC, REGARDLESS OF ORIENTATION.
C  SDEL1 - SIZE OF CHANGE IN SIGN INTERVAL.
C  SDEL2 - PREVIOUS VALUE OF SDEL1.
C  SDEL3 - PREVIOUS VALUE OF SDEL2.
C  SDEL4 - PREVIOUS VALUE OF SDEL3.
C  STEP  - THE NEW ROOT IS COMPUTED AS A CORRECTION TO B OF THE
C          FORM B(NEW)=B(OLD)+STEP.
C  TOLER - A NUMBER WE ACCEPT AS 'SMALL' WHEN EXAMINING INTERVAL
C          SIZE OR STEP SIZE.  TOLER=2.0*EPMACH*ABS(B) + EPMACH IS
C          A MINIMUM BELOW WHICH WE WILL NOT ALLOW SUCH VALUES TO FALL.
C  THIS SUBROUTINE IS CALLED BY
C    PITCON
C  AND CALLS
C    FORTRAN ABS
C    FORTRAN SIGN
C
C**********************************************************************
C
      REAL A,B,C,FA,FB,FC,STEP,TOLER,P,Q,R,S
      COMMON /TOL/    EPMACH,EPSATE,EPSQRT
C
C  SEGMENT 1:  FIRST CALL HANDLED SPECIALLY.  DO BOOKKEEPING.
C
C  SET CERTAIN VALUES ONLY FOR INITIAL CALL WITH KOUNT=0
C
      IF (KOUNT.GT.0) GO TO 10
      IF (FA.GT.0.0.AND.FC.GT.0.0) GO TO 110
      IF (FA.LT.0.0.AND.FC.LT.0.0) GO TO 110
      KOUNT=1
      SDEL1=2.0*ABS(C-A)
      SDEL2=2.0*SDEL1
      SDEL3=2.0*SDEL2
      B=C
      FB=FC
      GO TO 20
C
C  ON EVERY CALL, INCREMENT COUNTER
C
   10 KOUNT=KOUNT+1
C
C  RETURN IF HIT MACHINE ZERO FOR F(B)
C
      IF(FB.EQ.0.0) GO TO 90
C
C  SEGMENT 2:  REARRANGE POINTS AND FUNCTION VALUES IF
C  NECESSARY SO THAT FB*FC.LT.0.0, AND SO THAT
C  ABS(FB).LT.ABS(FC)
C
      IF((FB.LE.0.0).AND.(FC.GT.0.0)) GO TO 30
      IF((FB.GT.0.0).AND.(FC.LE.0.0)) GO TO 30
C
C  FB AND FC ARE OF SAME SIGN.
C  (ROOT CHANGED SIGN)
C  OVERWRITE C WITH VALUE OF A
C
   20 C=A
      FC=FA
```

```
C
C   IF NECESSARY, SET A:=B, B:=C, C:=B
C   TO ENSURE THAT ABS(FB).LE.ABS(FC)
C
    30 IF(ABS(FC).GE.ABS(FB)) GO TO 40
       A=B
       B=C
       C=A
       FA=FB
       FB=FC
       FC=FA
C
C   SEGMENT 3:  CHECK FOR ACCEPTANCE BECAUSE OF SMALL INTERVAL
C   CURRENT CHANGE IN SIGN INTERVAL IS (C,B) OR (B,C).
C
    40 TOLER=2.0*EPMACH*ABS(B)+EPMACH
       HALFBC=0.5*(C-B)
       SDEL4=SDEL3
       SDEL3=SDEL2
       SDEL2=SDEL1
       SDEL1=ABS(C-B)
       IF(ABS(HALFBC).LE.TOLER) GO TO 100
C
C   SEGMENT 4:  COMPUTE NEW APPROXIMANT TO ROOT OF THE FORM
C   B(NEW)=B(OLD)+STEP.
C   METHODS AVAILABLE ARE LINEAR INTERPOLATION
C   INVERSE QUADRATIC INTERPOLATION
C   AND BISECTION.
C
       IF(ABS(FB).GE.ABS(FA))GO TO 70
       IF(A.NE.C) GO TO 50
C
C   ATTEMPT LINEAR INTERPOLATION IF ONLY TWO POINTS AVAILABLE
C   COMPUTE P AND Q FOR APPROXIMATION B(NEW)=B(OLD)+P/Q
C
       IFLAG=2
       S=FB/FA
       P=2.0*HALFBC*S
       Q=1.0-S
       GO TO 60
C
C   ATTEMPT INVERSE QUADRATIC INTERPOLATION IF THREE POINTS AVAILABLE
C   COMPUTE P AND Q FOR APPROXIMATION B(NEW)=B(OLD)+P/Q
C
    50 IFLAG=3
       S=FB/FA
       Q=FA/FC
       R=FB/FC
       P=S*(2.0*HALFBC*Q*(Q-R)-(B-A)*(R-1.0))
       Q=(Q-1.0)*(R-1.0)*(S-1.0)
C
C   CORRECT THE SIGNS OF P AND Q
C
    60 IF(P.GT.0.0)Q=-Q
       P=ABS(P)
C
C   IF P/Q IS TOO LARGE, GO BACK TO BISECTION
C
       IF(8.0*SDEL1.GT.SDEL4) GO TO 70
       IF (P.GE.1.5*ABS(HALFBC*Q)-ABS(TOLER*Q)) GO TO 70
       STEP=P/Q
       GO TO 80
C
C   PERFORM BISECTION:
C    IF ABS(FB).GE.ABS(FA)
C    OR INTERPOLATION IS UNSAFE (P/Q IS LARGE)
C    OR IF THREE CONSECUTIVE STEPS HAVE NOT DECREASED
C       THE SIZE OF THE INTERVAL BY A FACTOR OF 8.0
C
    70 IFLAG=1
       STEP=HALFBC
       GO TO 80
```

```
C
C  SEGMENT 5:  VALUE OF STEP HAS BEEN COMPUTED.
C  UPDATE INFORMATION: A:=B, FA:=FB, B:=B+STEP.
C  CHANGE IN SIGN INTERVAL IS NOW (A,C) OR (C,A).
C
   80 A=B
      FA=FB
      IF(ABS(STEP).LE.TOLER) STEP=SIGN(TOLER,HALFBC)
      B=B+STEP
      RETURN
C
C  SPECIAL RETURNS
C
C   INPUT POINT B IS EXACT ROOT
C
   90 IFLAG=0
      RETURN
C
C  CHANGE IN SIGN INTERVAL IS OF SIZE LESS THAN 4*EPMACH*ABS(B)+EPMACH
C  INTERVAL RETURNED AS (B,C) OR (C,B).
C  ACCEPT B AS ROOT WITH RESIDUAL F(B) STORED IN FB.
C
  100 IFLAG=-1
      A=B
      FA=FB
      RETURN
C
C  CHANGE OF SIGN CONDITION VIOLATED
C
  110 IFLAG=-2
      KOUNT=0
      RETURN
C
C***********************************************************************
C
      END
```

```
      SUBROUTINE SOLVE(NVAR,X,Y,IP,DETA,IEXP,IERR,ICALL,IMOD,FPRYM,
     1 IPVT)
C
C***********************************************************************
C
C  THIS SUBROUTINE IS CALLED BY
C    CORECT
C    TANGNT
C  AND CALLS
C    FORTRAN ABS
C    LINPAK SGEFA
C    LINPAK SGESL
C    USER FPRIME
C
C  THIS SUBROUTINE SOLVES THE LINEAR SYSTEM  DFA(X,IP)*YOUT = YIN
C  WHERE DFA(X,IP) IS THE (NVAR)X(NVAR) MATRIX WHOSE FIRST NVAR - 1
C  ROWS ARE THE JACOBIAN COMPUTED BY FPRIME, AND WHOSE LAST
C  ROW IS ALL 0 EXCEPT FOR A 1 IN THE IP-TH COMPONENT.
C
C  YIN IS THE NVAR COMPONENT VECTOR Y ON INPUT, AND THE SOLUTION
C  VECTOR YOUT IS RETURNED IN Y ON OUTPUT AFTER A SUCCESSFUL
C  SETUP AND SOLUTION.
C
C  **NOTE** SUBROUTINE SOLVE USES FULL MATRIX STORAGE TO SOLVE THE
C  LINEAR SYSTEM.  THE USER MAY WISH TO REPLACE THIS ROUTINE WITH
C  ONE MORE SUITED TO HIS PROBLEM.
C
C  DETA    BINARY MANTISSA OF THE DETERMINANT OF JACOBIAN DFA(X,IP)
C  IEXP    BINARY EXPONENT OF THE DETERMINANT OF JACOBIAN DFA(X,IP)
C  IMOD    NEWTON METHOD FLAG.
C          IMOD=0, JACOBIAN IS TO BE EVALUATED FOR EVERY CORRECTOR STEP
C                  AND EVERY TANGENT CALCULATION
C          IMOD=1, JACOBIAN IS TO BE EVALUATED ONLY FOR FIRST CORRECTOR
C                  STEP, AND EVERY TANGENT CALCULATION
C  ICALL   SET UP FLAG.
C          IF (ICALL.EQ.0.AND.IMOD.NE.0) DON'T RE-EVALUATED JACOBIAN
C  INFO    OUTPUT FROM SGEFA.  IF INFO.NE.0, SGEFA FOUND A ZERO
C          PIVOT WHEN ELIMINATING INFO-TH VARIABLE.
C  IERR    RETURN FLAG, 0 MEANS SUCCESSFUL SOLUTION, 1 MEANS FAILURE
C  NVAR    THE NUMBER OF VARIABLES IN THE NONLINEAR SYSTEM
C  X       THE POINT AT WHICH TO EVALUATE FPRYM
C  Y       THE RIGHT HAND SIDE ON INPUT, THE SOLUTION
C          ON OUTPUT
C  FPRYM   ARRAY WHERE DFA(X,IP) IS TO BE STORED.
C  IPVT    INTEGER WORK SPACE FOR PIVOT ROW SWITCHES DEMANDED BY SGEFA
C  IP      THE VARIABLE USED IN THE AUGMENTING EQUATION THAT IS OF THE
C          FORM X(IP)=B.  HENCE THE LAST ROW OF DFA(X,IP) IS ALL
C          ZERO EXCEPT FOR A 1.0 IN THE IP-TH COLUMN.
C
C***********************************************************************
C
      REAL X(NVAR),Y(NVAR),FPRYM(NVAR,NVAR)
      INTEGER IPVT(NVAR)
      COMMON /COUNT2/ IFEVAL,IPEVAL,ISOLVE,NRED,NRDSUM,KN,KNSUM
C
C  DEPENDING ON VALUES OF ICALL AND IMOD, EITHER SET UP
C  AUGMENTED JACOBIAN, DECOMPOSE INTO L-U FACTORS, AND GET DETERMINANT,
C  OR USE CURRENT FACTORED JACOBIAN WITH NEW RIGHT HAND SIDE.
C
      IF (ICALL.EQ.0.AND.IMOD.NE.0) GO TO 50
      CALL FPRIME(NVAR,X,FPRYM)
      IPEVAL=IPEVAL+1
      DO 10 I=1,NVAR
   10     FPRYM(NVAR,I)=0.0
      FPRYM(NVAR,IP)=1.0
C
C  CARRY OUT IN CORE LU DECOMPOSITION OF NVAR BY NVAR MATRIX
C  AND USE PIVOT INFORMATION TO COMPUTE DETERMINANT
```

```
C
      CALL SGEFA(FPRYM,NVAR,NVAR,IPVT,INFO)
      DETA=1.0
      IEXP=0
      TWO=2.0
      DO 40 I=1,NVAR
            IF (IPVT(I).NE.I) DETA=-DETA
            DETA=FPRYM(I,I)*DETA
            IF (DETA.EQ.0.0) GO TO 60
   20       IF (ABS(DETA).GE.1.0) GO TO 30
            DETA=DETA*TWO
            IEXP=IEXP-1
            GO TO 20
   30       IF (ABS(DETA).LT.TWO) GO TO 40
            DETA=DETA/TWO
            IEXP=IEXP+1
            GO TO 30
   40       CONTINUE
      IF (INFO.NE.0) GO TO 60
C
C  USING L-U FACTORED MATRIX, SOLVE SYSTEM USING FORWARD-BACKWARD
C  ELIMINATION, AND OVERWRITE RIGHT HAND SIDE WITH SOLUTION
C
   50 CALL SGESL(FPRYM,NVAR,NVAR,IPVT,Y,0)
      IERR=0
      RETURN
   60 IERR=1
      INFO=0
      RETURN
C
C***********************************************************************
C
      END
```

REFERENCES

1. J. P. Abbott, An efficient algorithm for the determination of certain bifurcation points, J. of Computational and Applied Math. 4, 1978, 19-27.

2. E. Allgower and K. Georg, Simplicial and continuation methods for approximating fixed points and solutions to systems of equations, SIAM Review 22, 1980, 28-85.

3. P. T. Boggs, The solution of nonlinear systems of equations by A-stable integration techniques, SIAM J. Num. Anal. 8, 1971, 767-785.

4. R. P. Brent, Algorithms for minimization without derivatives, Prentice Hall, Englewood Cliffs, NJ, 1973.

5. M. A. Crisfield, A fast incremental/iterative solution procedure that handles snap-through, Computers and Structures 13, 1981, 55-62.

6. D. F. Davidenko, On a new method of numerical solution of systems of nonlinear equations, Dokl. Akad. Nauk USSR 88, 1953, 601-602.

7. C. Den Heijer and W. C. Rheinboldt, On steplength algorithms for a class of continuation methods, SIAM J. Num. Anal. 18, 1981, 925-947.

8. F. Ficken, The continuation method for functional equations, Comm. Pure Appl. Math. 4, 1951, 435-456.

9. F. Freudenstein and B. Roth, Numerical solution of systems of nonlinear equations, J. ACM 10, 1963, 550-556.

10. H. B. Keller, Numerical solution of bifurcation and nonlinear eigenvalue problems, in "Applications of Bifurcation Theory", ed. by P. Rabinowitz, Academic Press, New York, NY, 1977, 359-384.

11. H. B. Keller, Global homotopies and Newton methods in "Recent Advances in Numerical Analysis", ed. by C. deBoor, G. H. Golub, Academic Press, New York, NY, 1978, 73-94.

12. A. D. Kerr and M. T. Soifer, The linearization of the prebuckling state and its effect on the determined instability load, Trans. ASME, J. of Applied Mech., V. 36, 1969, 775-783.

13. W. Kizner, A numerical method for finding solutions of nonlinear equations, SIAM J. Appl. Math. 12, 1964, 424-428.

14. M. Kubicek, Algorithm 502, Dependence of solution of nonlinear systems on a parameter, ACM-TOMS 2, 1976, 98-107.

15. M. Kubicek, M. Holodniok and I. Marek, Numerical solution of nonlinear equations by one-parameter imbedding methods, Num. Functional Anal. and Optim. 3, 1981, 223-264.

16. S. T. Mau and R. H. Gallagher, A finite element procedure for nonlinear prebuckling and initial postbuckling analysis, NASA Contractor Report, NASA-CR_1936, January 1972.

17. J. W. Milnor, Topology from the differential viewpoint, Univ. of Virginia Press, Charlottesville, VA, 1965.

18. R. K. Mehra, W. C. Kessel and J. V. Carroll, Global stability and control analysis of aircraft at high angles of attack, ONR Report-CR-215-248 1,2,3, June 1977,78,79.

19. R. Menzel and H. Schwetlick, Zur Lösung parameter-abhängiger nichtlinearer Gleichungen mit singulären Jacobi-Matrizen, Num. Math. 30, 1978, 65-79.

20. G. Moore and A. Spence, The calculation of turning points of nonlinear equations, SIAM J. Num. Anal. 17, 1980, 567-576.

21. J. T. Oden, Finite elements of nonlinear continua, McGraw Hill, New York, 1972.

22. G. Pönisch and H. Schwetlick, Computing turning points of curves implicitly defined by nonlinear equations depending on a parameter, Computing 26, 1981, 107-121.

23. T. Poston and I. Stewart, Catastrophe theory and its applications, Pitman Publ. Ltd, London, 1978.

24. W. C. Rheinboldt, An adaptive continuation process for solving systems of nonlinear equations, Polish Academy of Science, Banach Ctr. Publ., Vol. 3, 1977, 129-142.

25. W. C. Rheinboldt, Numerical methods for a class of finite dimensional bifurcation problems, SIAM J. Num. Anal. 15, 1978, 1-11.

26. W. C. Rheinboldt, Solution fields of nonlinear equations and continuation methods, SIAM J. Num. Anal. 17, 1980, 221-237.

27. W. C. Rheinboldt, Numerical analysis of continuation methods for nonlinear structural problems, Computers and Structures 13, 1981, 103-114.

28. W. C. Rheinboldt, Computation of critical boundaries on equilibrium manifolds, Univ. of Pittsburgh, Inst. f. Comp. Math. and Appl., Techn. Rept. ICMA-80-20, 1980; SIAM J. Num. Anal., 1981, in press.

29. E. Riks, An incremental approach to the solution of snapping and buckling problems, Int. J. Solids and Structures 15, 1979, 524-551.

30. A. A. Schy and M. E. Hannah, Prediction of jump phenomena in roll-coupled maneuvers of airplanes, J. of Aircraft 14, 1977, 375-382.

31. M. J. Sewell, On the connection between stability and the shape of the equilibrium surface, J. Mech. Phys. Solids 14, 1966, 203-230.

32. M. J. Sewell, Some global equilibrium surfaces, Int. J. Mech. Engng. Educ. 6, 1978, 163-174.

33. R. Seydel, Numerical computation of branch points in nonlinear equations, Numer. Math. 33, 1979, 339-352.

34. R. B. Simpson, A method for the numerical determination of bifurcation states of nonlinear systems of equations, SIAM J. Num. Anal. 12, 1975, 439-451.

35. H. J. Wacker (editor), Continuation methods, Academic Press, New York, NY, 1978.

36. A. C. Walker, A non-linear finite element analysis of shallow circular arches, Int. f. Solids and Structures 5, 1969, 97-107.

37. L. T. Watson, A globally convergent algorithm for computing fixed points of C^2-maps, Appl. Math. and Comp. 5, 1979, 297-311.

38. L. T. Watson and D. Fenner, Algorithm 555: Chow-Yorke algorithm for fixed points or zeros of C^2-maps, ACM Trans. Math. Software 6, 1980, 252-260.

39. J. W. Young, A. A. Schy and K. G. Johnson, Prediction of jump phenomena in aircraft maneuvers, including nonlinear aerodynamic effects, J. of Guidance and Control 1, 1978, 26-31.

DATE

FILMED

8