و جمع ا







Technical Report ICMA-81-30

1

October 1981

A Program for a Locally-Parametrized Continuation Process^{*)}

-13.1

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.

DISTRIBUTION	C. ATELESAT A
Apercyere : Distanction	a Contraction

*) 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.

111210

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 increme 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 liter-

ature 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, $0 \le t \le 1$. (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), $0 \le t \le 1$, 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

(2.1)
$$G(y,p) = 0$$

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

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

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

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: $R^r \rightarrow R^{r-1}$ and wish to compute the subset of E(G) defined by the augmented equations

(2.3)
$$G(y,p) = 0,$$

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)
$$Fx = 0$$

with a user-specified mapping F: $\mathbb{R}^n \to \mathbb{R}^{n-1}$. 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:

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

(2.5) (ii) The derivative DF(x) of F is locally lipschitzian on R''.

(iii) The regularity set $R(F) = \{x \in R^n; rank DF(x) = n-1\}$ 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) T:
$$R(F) \rightarrow R^{n}$$
, $DF(x)Tx = 0$, $||Tx||_{2} = 1$, $det \begin{pmatrix} DF(x) \\ (T_{x})^{T} \end{pmatrix} > 0$

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

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 $E(F) \wedge R(F)$. For the discussion it is useful to consider a parametrization by arclength of $E^*(F,x^0)$, that is, a function $x: J \rightarrow E^*(F,x^0)$ which maps some interval $J \leftarrow R^1$ 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,..., corresponding to some sequence $0 = s_0 < s_1 < s_2 < ...$ of arclength values. Note, however, that in general the values $s_1, s_2,...$ 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 datahandling 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)
$$\det \begin{pmatrix} DF(x) \\ (e^{i})^{T} \end{pmatrix} = [(e^{i})^{T}Tx] \det \begin{pmatrix} DF(x) \\ (Tx)^{T} \end{pmatrix}, \forall x \in R(F), i = 1,...,n,$$

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

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

is uniquely defined. Evidently, then

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

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

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

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 $x^j: J_j \subseteq R^1 \rightarrow R(F)$, $j = 1, \ldots, 4$, 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 $\lim x^1(s) = -\lim x^2(s)$ 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.

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

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

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 , $1 \le i_k \le n$, 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

(i) $|(e^{j_1})^T T x^k| < |(e^{j_1})^T T x^{k-1}|$,

(3.7) (ii) $|(e^{j_2})^T T x^k| > |(e^{j_2})^T T x^{k-1}|$,

(iii)
$$|(e^{J_2})^T T_x^k| \ge \mu |(e^{J_1})^T T_x^k|,$$

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

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

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^7 , y^2 ,... 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

 $(i) \ || \, \hat{F} y^{j} || \geq \theta || \, \hat{F} y^{j-1} || \quad \text{for some} \quad j \geq 1,$

- (3.9) (ii) $||y^{j}-y^{j-1}|| \ge \theta ||y^{j-1}-y^{j-2}||$ for some $j \ge 2$,
 - (iii) $j \ge j_{max}$.

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:

(i) $||\hat{F}y^{j}|| \le 8 \varepsilon_{mach}$ for some $j \ge 0$,

(3.10)

(ii) $(||Fy^{j}|| \leq \varepsilon_{abs})$ and $(||y^{j}-y^{j-1}|| \leq \varepsilon_{abs} + \varepsilon_{rel}||y^{j}||)$ for some $j \geq 1$.

The tolerances ε_{abs} , ε_{rel} are user specified and ε_{mach} is the smallest floating point number such that 1. = 1. + ε_{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,..., approximating the continuation curve $x: J + E^*(F, x^0)$ 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, ... 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)
$$0 \le e_{i+1} \le \lambda e_i, i = 0, 1, ...$$

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)
$$\tilde{\lambda} = \tilde{\omega}^{1/(i^{*}-1)}, \quad \tilde{\omega} = \frac{||y^{i^{*}}-y^{i^{*}-1}||}{||y^{i^{*}}-y^{0}||}, \quad i^{*} \geq 2,$$

represents a computable estimate of λ .

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

(4.3)
$$\delta_{k} = ||\hat{x}^{k} - x^{k+1}|| = ||y^{0} - y^{i^{*}}||$$

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_0 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)
$$\theta_{k} = \tilde{\lambda}^{i^{\star}-m^{\star}} \equiv \tilde{\omega}^{(i^{\star}-m^{\star})/(i^{\star}-1)}$$

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

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,..., satisfy

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

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)
$$\varepsilon_{i} \leq n_{i} \equiv \phi^{i}(n_{o}) \equiv \frac{3}{1+2 \cosh 2^{i} \alpha}, \quad i = 0, 1, ..., \quad n_{o} = \varepsilon_{o},$$

where α is the unique positive solution of

(4.8)
$$\psi(\alpha) = n_0, \quad \psi(\alpha) = \frac{3}{1+2 \cosh \alpha}.$$

Moreover, for any ω , $0 < \omega < 1$, and $i \ge 2$ 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)
$$\tilde{\omega} = \frac{||y^{i*}-y^{i*-1}||}{||y^{i*}-y^{0}||} = \frac{e_{i*-1}}{e_{0}} = \frac{\varepsilon_{i*-1}}{\varepsilon_{0}} \leq \frac{n_{i*-1}}{n_{0}}$$

and compute with this $\tilde{\omega}$ the solution $\tilde{\alpha}$ of (4.9) which gives the estimate $\tilde{n}_{0} = \psi(\tilde{\alpha})$ of ε_{0} . Now we proceed as before and obtain the factor

(4.11)
$$\theta_{k}^{\star} = \frac{\delta_{k}^{\star}}{\delta_{k}} = \frac{\tilde{n}_{0}^{\star}}{\tilde{n}_{0}}$$

for the ideal starting error by determining the unique solution $~\tilde{n}_0^{\,\prime},~0<\tilde{n}_0^{\,\prime}<1,$ of

(4.12)
$$\phi^{m^*}(\tilde{n}_0) = \phi^{i^*}(\tilde{n}_0).$$

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 $0.125 \leq \theta_k \leq 8$.

	• ω̃ε[a	,b]	
1-	a	b	[™] k
	0.8735115	1	1
2	0.1531947	0.8735115	0.9043128 - 0.7075675 ln ω
2	0.03191815	0.1531947	-4.667383 - 3.677482 ln ω
	0	0.03191815	8
	0.4677788	1	1
2	0.6970123(-3)	0.4677788	$0.8516099 - 0.1953119 \ln \omega$
3	0.1980863(-5)	0.6970123(-3)	-4.830636 - 0.9770528 ln ω
	0	0.1980863(-5)	8
4	0	1	1
_	0.3339946(-10)	1	1.040061 + 0.03793395 ln ω
5	0	0.3339946(-10)	0.125
	0.1122789(-8)	1	1.042177 + 0.04450706 ln ω
0	0	0.1122789(-8)	0.125
<u>></u> 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)
$$q(t) = x^{k} + t Tx^{k} + \frac{1}{2}t^{2}w^{k}, w^{k} = \frac{1}{\Delta s_{k}}(Tx^{k}-Tx^{k-1}), \Delta s_{k} = ||x^{k}-x^{k-1}||_{2}$$

for which

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

Since

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

the quantity

(4.16)
$$||w^{k}||_{2} = \frac{2}{\Delta s_{k}} |\sin \frac{1}{2} \alpha_{k}|, \quad \alpha_{k} = \arccos ((Tx^{k})^{T} Tx^{k-1})$$

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)
$$\gamma_{k}^{\text{tent}} = ||w^{k}||_{2} + \frac{\Delta s_{k}}{\Delta s_{k} + \Delta s_{k-1}} (||w^{k}||_{2} - ||w^{k-1}||_{2})$$

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

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

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

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

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

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)
$$\varepsilon_{k} = \begin{cases} \varepsilon_{\min} \Delta s_{k} & \text{if } \delta_{k}^{*} \leq \varepsilon_{\min} \Delta s_{k} \\ \Delta s_{k} & \text{if } \delta_{k}^{*} \geq \Delta s_{k} \\ \delta_{k}^{*} & \text{otherwise} \end{cases}$$



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

(4.21)
$$h_{k}^{(1)} = \sqrt{\frac{2\varepsilon_{k}}{\gamma_{k}}}.$$

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)
$$h_{k}^{(2)} = h_{k}^{(1)} [1 + \frac{h_{k}^{(1)}}{2\Delta s_{k}} (1 - \frac{(e^{i})^{T} Tx^{k-1}}{(e^{i})^{T} Tx^{k}})].$$

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

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

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 $\tilde{x}_i = XIT$. 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:

<u>Target points</u>: 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)
$$y(t) = (1-t)x^{k-1} + tx^k, \quad 0 \le t \le 1,$$

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.

<u>Limit points</u>: 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 $sign(e^i)^T Tx^{k-1} \neq sign(e^i)^T Tx^k$ for $i = LIM (\neq 0)$ 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 $x: [s_{k-1}, s_k] \neq E^*(F, x^0)$ 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 $\phi: [0,1] \neq [s_{k-1}, s_k]$ 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) g:
$$[0,1] \rightarrow R^{1}$$
, g(t) = $(e^{i})^{T} T_{X}(\phi(t))$, $0 \le t \le 1$,

and our problem is to determine a zero of g. Since by assumption sign $g(0) \neq sign g(1)$, 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 $g(t) = (e^{\hat{i}})^T Tx$. 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.

<u>Example 1</u>. 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

(6.1)
$$F_{x} = \begin{pmatrix} x_{1} - x_{2}^{3} + 5x_{2}^{2} - 2x_{2} + 34x_{3} - 47 \\ \\ x_{1} + x_{2}^{3} + x_{2}^{2} - 14x_{2} + 10x_{3} - 39 \end{pmatrix}, \quad \forall x \in \mathbb{R}^{3}.$$

For the starting point $x^{0} = (15, -2, 0)^{T}$ the solution cyree 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 $h_0 = 0.3$ 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.

		Continuation p	oint	Contin	Total	ļ
Step	×ı	×2	×3	Variable	Steps	Comments
Ũ	15.000	-2.00000	0.00000	×3	-	
ו	14.705	-1.9421	0.065381	۲×	2	
2	14.285	-1.7291	0.26874	×3	3	
3	16.906	-1.2094	0.54684	×2	2	
4	24.918	-0.59908	0.55514	۲	3	
. 5	48.974	0.71803	-0.080758	۲	3	
6	57.928	1.2846	-0.40736	۲	4	Step red.
7	60.052	1.5709	-0.54035	۲×	4	Step red.
8	61.666	2.0010	-0.66683	×2	2	
9	-5.1039	4.1510	1.3464	×2	2	Target passed
				TOTAL	25	
			Computatio	n of target	4	

Table 2

		Continuation p	oint	Contin	Total Correct	
Step	×ı	×2	×3	Variable	Steps	Comments
0	15.000	-2.0000	0.00000	×3	-	
1	14.710	-1.9421	0.065381	×1	3	
2	14.285	-1.7291	0.26874	×3	4	
3	16.906	-1.2094	0.54685	×2	1	
4	24.918	-0.59906	0.55514	ר [×] ו	6	
5	48.975	0.71810	-0.080804	۲	5	
6	57.289	1.2847	-0.40742	×ı	6	Step red.
7	60.053	1.5711	-0.54042	۲	5	Step red.
8	61.666	2.0013	-0.66689	×2	2	
9	-4.4239	4.1413	1.3229	×2	1	Target passed
				TOTAL	33	
			Computatio	n 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.

<u>Example 2</u>. 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) $Fx \equiv Ax + \phi(x) = 0, \quad \forall x \in \mathbb{R}^8$

where

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

and

$$\phi(\mathbf{x}) = \begin{pmatrix} -0.727 \ \mathbf{x}_2 \mathbf{x}_3 + 8.39 \ \mathbf{x}_3 \mathbf{x}_4 - 684.4 \ \mathbf{x}_4 \mathbf{x}_5 + 63.5 \ \mathbf{x}_4 \mathbf{x}_7 \\ 0.949 \ \mathbf{x}_1 \mathbf{x}_3 + 0.173 \ \mathbf{x}_1 \mathbf{x}_5 \\ -0.716 \ \mathbf{x}_1 \mathbf{x}_2 - 1.578 \ \mathbf{x}_1 \mathbf{x}_4 + 1.132 \ \mathbf{x}_4 \mathbf{x}_7 \\ - \ \mathbf{x}_1 \mathbf{x}_2 \\ \mathbf{x}_1 \mathbf{x}_4 \end{pmatrix}$$

Figure 3 shows some solution curves on the three-dimensional equilibrium surface in \mathbb{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



×1 ×2 ×3 ×4 ×5 ×6 ×7 1 2.9649 0.82557 0.073661 0.041309 0.26735 -0.05 0.50481 2 2.8174 -0.17629 -0.071476 -0.008 -0.20497 0.089926 0.026429 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 -0.68972 2.2992 -1.4102 -0.061849 -0.079009 -0.58630 0.1 4.4565 -4.4909 0.33091 -1.0857 0.1 10.0212 8 1.6164

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

Table 5

<u>Example 3</u>. 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 $EAR(H/R)^2$ --



is given by

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

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

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

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

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

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

,

Sector Sector

	SURROUTINE PITCON (NVAR + 1 M+ IT + XIT + KSTEP + IPC + H+ TRET + THOD + TPVT +	PTCN0001
r	I HRAZZHRINZHFAULZARSERKZE (* .KRZKWUKKZISIZE)	FILN0002
č*:	***************************************	XPTCNC004
Ċ		PTCN0005
Č.	PITCON.FOR	PTCN0006
č	00 007 4004 UFROTON OF ATTOON	PTCN0007
C C	27 UCH 1981 VERSION OF PITCONDER CONTINUATION PACKAGE	PTCN0008
č	THIS VERSION USES SINGLE PREFISION AND FULL NATRY STORAGE.	PTCN0007
č	THE COMMENT OF CARGE TREACED THE THE THE THE COMPLET	PTCN0011
Ĉ		PTCN0012
Č	THIS PACKAGE WAS PREPARED WITH THE PARTIAL SUPPORT OF	PTCN0013
Č.	THE NATIONAL SCIENCE FOUNDATION; UNDER GRANT MCS-78-05299;	PTCN0014
L C		PICNO015
č	INSTITUTE FOR COMPUTATIONAL MATHEMATICS AND APPLICATIONS	PTCN0017
Ē	DEPARTMENT OF MATHEMATICS AND STATISTICS,	PTCN0018
Ç	UNIVERSITY OF PITTSBURGH, PITTSBURGH, PA 15261.	PTCN0019
č		PTCN0020
č	AN CARLIER HERCTON OF THE DACKAGE HAC HOLTTEN IN COOSEDATION HITH	PTCN0021
ř	AN EARLIER VERSION OF THE PACKAGE WAS WRITTEN IN COUPERATION WITH	PTUN9922
č	GEORGE D RYRNE.	PTCN0023
č	COMPUTING TECHNOLOGY AND SERVICES	PTCN0025
Ç	EXXON RESEARCH AND ENGINEERING COMPANY	PTCN0026
č	LINDEN, NEW JERSEY, 07036	PTCN0027
Ľ.		PTCN0028
č	THIS PACKAGE COMPLITES POINTS ALONG A SOLUTION CURVE OF AN	PTCN0027
č	UNDERDETERMINED SYSTEM OF NONLINEAR EQUATIONS OF THE FORM FX=0.	PTCN0031
Ĉ	THE CURVE IS SPECIFIED TO PASS THROUGH A GIVEN STARTING SOLUTION	PTCN0032
Ç	X OF THE SYSTEM. HERE X DENOTES A REAL VECTOR OF NVAR	FTCN0033
č	COMPONENTS AND FX A REAL VECTOR OF NVAR-1 COMPONENTS.	PTCN0034
č	NUKMALLI LALK VALL VU FIVUN FRUUULAS A NEW FUINT FURTHER ALUNG	PICN0035
č	AN OPTION ALLOWS THE SEARCH FOR AND COMPUTATION OF TARGET POINTS.	PTCN0037
č	THAT IS, SOLUTION POINTS X FOR WHICH X(IT) = XIT FOR SOME USER	PTCN0038
Ċ	SPECIFIED VALUES OF IT AND XIT.	PTCN0039
Č.	A FURTHER OPTION ALLOWS THE SEARCH FOR AND COMPUTATION OF LIMIT	PTCN0040
Č,	PUINTS FOR SPECIFIED COURDINATE LIMP THAT ISP SOLUTION POINTS FOR	PTCN0041
ř	WALCH THE LANTIN CONFUMENT OF THE THINDERT VELTOK AD ARAU.	PTCN0042
č		PTCN0044
č	EXPLANATIONS OF THE ALGORITHMS USED IN THIS PACKAGE MAY	PTCN0045
C	BE FOUND IN THE FOLLOWING REFERENCES:	PTCN0046
Č		PTCN0047
č		PTCN0048
ř	WERNER RHEINBULLUT Soutton stein af nami infar fonations and continuation methods	PTCN0047
č	SIAN JOURNAL OF NUMERICAL ANALYSIS, 17, 1980, PP 221-237	PTCN0051
č		PTCN0052
Ĉ	COR DEN HEIJER AND WERNER RHEINBOLDT,	PTCN0053
Č.	ON STEPLENGTH ALGORITHMS FOR A CLASS OF CONTINUATION METHODS,	PTCN0054
Č.	SIAN JOURNAL OF NUMERICAL ANALYSIS 18, 1981, PP 925-947	PTCN0055
ř	HERNER RHEINRALAT.	PTCN0006
č	NUMERICAL ANALYSIS OF CONTINUATION METHODS FOR NONLINEAR	PTCN0058
Ĉ	STRUCTURAL PROBLEMS,	PTCN0059
č	COMPUTERS AND STRUCTURES, 13, 1981, PP 103-114	FTCN0060
č		PTCN0061
ř	NUTLINE OF THE MATHEMATICAL PROCEDURE	PTCN0062
č	THE FRAME OF THE PERTINETER FOR FULL	PTCNAAA
č		PTCN0065
C	THE FUNCTION F DEFINING THE SYSTEM OF EQUATIONS IS SUPPOSED TO	PTCN0066
č	BE CONTINUOUSLY DIFFERENTIABLE. THE REGULARITY SET R(F) OF F IS	PTCN0067
Č.	THE SET OF ALL POINTS & IN NVAR-OTMENSTONAL SPACE WHERE THE	FTCN0068
ř	TA RE IN RIEL, FOR ANY INDEX ID WITH LIE TRUE NUAR, LET	FILMOUST FTENCOTO

FA(X, (P) BE THE FUNCTION OBTAINED BY AUGMENTING F WITH PTCN0071 С AN AVAR-TH SCALAR FUNCTION X(IP)-B FOR SOME NUMBER B. FOR ANY X IN R(F) THERE IS AT LEAST ONE INDEX IP SUCH THAT THE DERIVATIVE DEFA(F) OF FA IS NONSINGULAR. WITH SUCH AN IP AND A DIRECTION DIR (=+1 OR -1), THE TANGENT OF FA, TAN(X) IS UNIQUELY DEFINED BY: PTCN0072 PTCN0073 #TCN0074 PTCN0075 FTENO PTONO07 TAN:= SOLUTION OF (DFA(X, JP)*TAN=E(NVAR) TAN:= TAN/(EUCLIDEAN NORM OF TAN) PTCNC073 PTCN0079 SN:= DIR*SIGN (DETERMINANT (DFA(X, IP))) С PTCNOO80 TAN:= SN#TAN PTCN0081 HERE E(I) IS THE I-TH BASIS VECTOR IN NVAR-SPACE, THE PROCESS USES A LOCAL PARAMETERIZATION OF THE CURVE, PTCN0032 PTCN0083 C NORMALLY THE CONTINUATION PARAMETER INDEX IP IS CHOSEN AS THAT INDEX FOR WHICH ABS(TAN(X)(IP)) IS MAXIMAL. BUT IN CASE OF CERTAIN CURVATURE CHANGES WHERE IT APPEARS THAT A LIMIT POINT FOR THIS CHOICE FOR IP IS APPROACHING, PTCN0084 BUT IN THE PTCN0085 PTCN0086 PTCN0087 OTHER CHOICES FOR IP MAY BE USED. PREDICTION TAKES PLACE ALONG THE EULER LINE X+H*TAN. THE STEPLENGTH ALGORITHM TAKES INTO ACCOUNT THE QUALITY OF THE CORRECTOR ITERATION AT THE LAST POINT AND A PREDICTION OF THE CHANGE IN CURVATURE. THE TANGENTIAL STEPSIZE USED IN PREDICTION IS CHOSEN SO AS TO ACHIEVE APPROXIMATELY THE PREDICTED SECANT STEPSIZE PTCNOORE PTCN0089 C PTCN0090 FTCN0091 Ĉ FTCN0092 PTCN0093 CHOSEN SO AS TO ACHIEVE APPROXIMATELY THE PREDICTED SECANT STEPSIZE PTCN0093 AFTER CORRECTION IS DONE. THE CORRECTOR ITERATION STARTS FROM THE PREDICTED POINT AND SOLVES PTCN0095 THE AUGMENTED SYSTEM FA(X,IP)=0 WITH THE VALUE OF THE SCALAR PTCN0095 B EQUAL TO THE IP-TH COMPONENT OF THE PREDICTED POINT. THE USER PTCN0097 CAN SPECIFY AS CORRECTOR ITERATION EITHER A FULL NEWTON PROCESS, PTCN0095 OR A MODIFIED NEWTON PROCESS WITH FIXED JACOBIAN DFA EVALUATED AT PTCN0097 THE PREDICTED POINT. 0 С PTCN0101 PTCNO1(OUTLINE OF THE COMPUTATIONAL ALGORITHM PTCN0103 FTCN0104 PTCN0105 Ĉ DURING THE FOLLOWING DESCRIPTION, WE WILL ASSUME THAT WE PTCN0106 HAVE ENTERED THE CONTINUATION LOOP WITH AN OLD POINT XL, A CURRENT POINT XC, THE TANGENT TL AT XL, AND CERTAIN SCALAR QUANTITIES ASSOCIATED WITH THESE VECTORS. WE WILL CHECK C PTCN0107 PTCN0108 Ć PTCN0109 FIRST FOR ANY TARGET OR LIMIT POINTS BETWEEN XL AND XC, THEN PROCEED TO COMPUTE A NEW CONTINUATION POINT XF. THESE NAMES ARE NOT IN PRECISE ACCORDANCE WITH THE STORAGE ARRANGEMENTS UNTIL THE END OF A CONTINUATION STEP. Ç FTCN0110 С PTCN0111 Ĉ PTCN0112 PTCN0113 PTCN0114 PTCN0115 FOR KSTEP.GT.0, THE CODE GOES TO STEP 2. ON THE FIRST CALL TO PITCON FOR A GIVEN PROBLEM (KSTEP=-1 PTCN0115 OR KSTEP=0) PROBLEM-DEPENDENT CONSTANTS ARE SET FTCN0117 OR KSTEP=0) PROBLEM-DEPENDENT CONSTANTS ARE SET FTCN0118 AND USER CONTROL PARAMETERS ARE LOADED OR DEFAULTS USED. PTCN0119 IF (KSTEP.E0.-1), THE PROBRAM PROCEEDS TO STEP 2. PTCN0120 IF (KSTEP.E0.-1), THE USER REDUESTS THAT THE INPUT STARTING PTCN0121 POINT XR RE CHECKED FOR THE CONDITION PTCN0122 ABS(F(XR)).LE. (ABSERR/2). IF THIS IS NOT THE CASE, NEWTON'SPTCN0123 METHOD IS APPLIED TO THE POINT XR UNTIL THE ERROR CONDITION PTCN0124 IS SATISFIED, OR A FAILURE HAS OCCURRED. AN UNIMPROVABLE PTCN0125 POINT RESULTS IN A RETURN OF IRET=-6. IF THE STARTING POINT XR WAS IMPROVED, THE PROGRAM RETURNS PTCN0127 WITH IRET=0 AND KSTEP=0. PTCN0126 STEP 1: IF THE STARTING POINT XR WAS IMPROVED, THE PROGRAM RETURNS WITH IRET=0 AND KSTEP=0. IF (KSTEP.EQ.O), THE CONTINUATION LOOP BEGINS WITH THE STARTING POINT XR STORED IN XL AND XC, THE STEPSIZE HTANCF SET TO THE IMPUT VALUE OF H, AND THE CONTINUATION PARAMETER SET TO THE IMPUT VALUE OF IPC. FOR KSTEP.GT.O, THESE QUANTITIES ARE COMPUTED AND UPCATED BY THE PROGRAM ITSELF. PTCN0128 PTCN0129 PTCN0130 PTCN0131 PTCN0132 PTCN0133 Č PTCN0134 0000000 FTCN0135 TARGET POINT CHECK. IF (IT.NE.O), A TARGET POINT IS DESIRED. THE VALUES OF XL(IT) AND XC(IT) ARE COMPARED TO XIT. IF THE TARGET POINT IS BETWEEN XL AND XC. THE PROGRAM PICNO138 COMPUTES THE TARGET POINT, SETS IRET=1, AND STEP 2:

RETURNS, TEMPORARILY INTERRUPTING NORMAL CONTINUATION.

С

C

£

35

PTCN0140

PTCN0141 TANGENT AND LOCAL CONTINUATION PARAMETER CALCULATION. IF THEPTCN0142 LOOP WAS SUSPENDED AT THE LAST CALL TO FUTCON TO ALLOW THE FTCN0143 RETURN OF A LIMIT POINT, THEN THE TANGENT HAS ALREADY BEEN PICN0144 STEP 3: FTCN0143 PTCN0144 FTCN0145 CALCULATED AND A LIMIT POINT CHECK IS SUPERFLUCUS, SO THE PROGRAM SKIPS TO STEP ັ5. THE PROUMER SATES TO STEP STANGENT PLANE AT XC IS COMPUTED, FICNO147 SUPPOSE THAT THE PREVIOUS CONTINUATION PARAMETER INDEX PICHO143 WAS IPL, WHERE ON THE FIRST STEP IPL IS USER-SUPPLIED, FICNO147 THE NEW TANGENT IS NORMALIZED, AND THE IPL-TH COMPONENT PICHO150 IS FORCED TO HAVE THE SAME SIGN AS THE IPL-TH COMPONENT PICHO152 OF THE PREVIOUS TANGENT (OR ON FIRST STEP: THE SAME SIGN AS THE USER INPUT DIRECTION DIR.) THEN THE LOCAL PTCN0153 PTCN0153 SIGN AS THE USER INFUT DIRECTION DIR.7 THEN THE CUCAL PTCN0153 CONTINUATION PARAMETER IPC IS DETERMINED. IPC IS SET TO THEFTCH0154 LOCATION OF THE LARGEST COMPONENT OF THE TANGENT VECTOR PTCN0155 UNLESS A LIMIT POINT FOR THIS CHOICE APPEARS TO 3E PTCN0156 APPROACHING, IN WHICH CASE THE LOCATION OF THE SECOND PTCN0155 LARGEST COMPONENT MAY BE TRIED. PTCN0156 ONCE IPC IS SET, THE RUANTITIES TCIPC, TCLIM, HSFCLC, ALPHLCPTCN0159 AND DIR ARE COMPUTED, WHOSE MEANINGS ARE EXPLAINED BELOW. PTCN0150 PTCN0161 LIMIT POINT CHECK. IF LIM.NE.O, THE LIM-TH COMPONENTS PTCN0162 OF THE OLD AND NEW TANGENTS ARE COMPARED. IF THESE DIFFER PTCN0163 IN SIGN, A LIMIT POINT LIES BETWEEN XL AND XC. THE PROGRAM PTCN0164 ATTEMPTS TO FIND THIS LIMIT POINT. IF FOUND, IT STORES PTCN0165 THE LIMIT POINT IN XR, THE TANGENT AT XR IN TL, SETS IRET=2,PTCN0160 STEP 4: AND RETURNS, TEMPORARILY INTERRUPTING THE NORMAL LOOF. PTCN0167 FTCN0168 STEP LENGTH COMPUTATION. THE PROGRAM COMPUTES HTANCF, THE STEPSIZE TO BE USED ALONG THE TANGENT TO ORTAIN THE PREDICTED POINT XPRED:=XC+HTANCF*TC, THE STARTING POINT FOR THE CORRECTOR PROCESS. IN COMPUTING HTANCF, CERTAIN CURVATURE AND STEPSIZE DATA ARE UPDATED. STEP 5: PTCN0169 PTCN0170 CCC PTCN0171 PTCN0172 FTCN0173 č c **PTCN0174** STEP 6: PREDICTION AND CORRECTION STEP. WITH THE PREDICTED POINT PTCN0175 XPRED=XC+HTANCF*TC AS A STARTING POINT, THE CORRECTOR PROCESS IS APPLIED TO CORRECT THE POINT UNTIL ARS(F(XCOR)).LE.ARSERR AND XSTEP, THE LAST CORRECTOR STEP, C PTCN0176 č PTCN0177 00000 ABS(F(XCOR)).LE.ABSERR AND XSTEP, THE LAST CORRECTOR STEP, SATISFIES XSTEP.LE.ABSERR+RELERR*ABS(XCOR). IF THE SIZE OF A CORRECTOR STEP IS TOO LARGE, OR IF A CORRECTION STEP INCREASES THE FUNCTION VALUE, OR THE MAXIMUM NUMBER OF STEPS ARE TAKEN WITHOUT CONVERGENCE, THE STEPSIZE HTANCF IS REDUCED AND THE CORRECTOR STEP IS ATTEMPTED AGAIN. IF THE STEPSIZE SHRINKS BELOW HMIN, THE PROGRAM SETS AN ERROR FLAG AND RETURNS. PTCN0178 PTCN0179 PTCN0180 PTCN0181 PTCN0182 PTCN0183 PTCN0184 PTCN0185 PTCN0185 STEP 7: STORING INFORMATION REFORE RETURN. AFTER A SUCCESSFUL PTCN0187 CONTINUATION STEP, THE PROGRAM REARRANGES ITS STORAGE SO THAT THE ENTRIES CORRESPONDING TO XC AND XF HOLD THE PROPER DATA, COMPUTES CORDIS, THE SIZE OF THE CORRECTION TO THE PREDICTED POINT, AND MODIFIES CORDIS TO A VALUE THAT WOULD CORRESPOND TO AN OPTIMAL NUMBER OF CORRECTOR STEPS. 00000 PTCN0138 PTCN0189 PTCN0190 PTCN0191 PTCN0192 PTCNOI93 PTCN0174 ON NORMAL RETURN, THE VECTOR XR (THE FIRST NVAR ENTRIES OF RWORK), CONTAINS A SOLUTION POINT ON THE CURVE F(XR)=0, AND IS EITHER A CONTINUATION POINT, A TARGET POINT, OR A LIMIT POINT, WHICH IS INDICATED BY THE VALUE OF IRET. IF JRET IS NEGATIVE, AN ERROR HAS OCCURRED. IF A LIMIT POINT IS DESTINGUED. THE TANGENT WEITHER TOWN TO A DATA THE TANK PTCN0195 PTCN0196 PTCN0197 PTCN0198 PTCN0199 RETURNED, THE TANGENT VECTOR AT THE LIMIT POINT IS CONTAINED IN THE PTCN0200 RETURNED, THE TANGENT VECTOR AT THE LIMIT POINT IS CONTAINED IN THE PTCN0200 LOCATION TL IN RWORK. ON FIRST CALL, THE USER MUST SET SOME OF THE PTCN0200 SCALAR PARAMETERS, AND THE STARTING POINT XR. THEREAFTER, ONLY IT PTCN0201 AND XIT SHOULD BE CHANGED BY THE USER DURING A PROBLEM RUN. PTCN0203 IF A NEW PROBLEM IS TO BE RUN (WHETHER A DIFFERENT FUNCTION, OR THEPTCN0204 SAME FUNCTION WITH DIFFERENT STARTING POINT OR ERROR CONTROLS) PTCN0205 THE PROGRAM MAY RE RESET BY USING KSTEP=-1 OK OK AT WHICH TIME THE SCALARS AND THE POINT XR MUST RE SET AGAIN. NOTE THAT IN THIS CASE THE STATISTICAL DATA IN THE COMMON BLOCKS /COUNTL/ AND /COUNT2/ WILL BE RESET TO 0 AS WELL. PTCN0206 PTCN0207 PTENOT 18 PTENOT09 C C FTENOTIO

C

C

CCCCCCCC

CCCCCC

ĉ

ē

С				9TEN0711
Ç	DEFINI	[1]	ONS AND DEFAULTS OF PITCON PARAMETERS	FTENOS12
C				PTCN0213
С				PTCN0214
С	NVAR	2	THE NUMBER OF VARIABLES IN THE NONLINEAR SYSTEM. NVAR IS	PTCN0215
Ç			THE DIMENSION OF THE PIVOT VECTOR IPVT, AND THE SIZE OF THE	PTCN0216
Ç			VECTORS XR, XC, XF, TL AND TC WHICH ARE CONTAINED IN RWORK.	PTCN0217
Č			RWORK ALSO CONTAINS STORAGE FOR THE MATRIX FPRYM WHICH	PTCN0218
Ľ			IS OF SIZE NVAR X NVAR.	PTCN0219
č			NVAR AUST BE GREATER THAN IF AND AUST NUT BE CHANGED DURING	PTCN0220
с С	1.754	_	THE LOURDE OF A PROBLEM RUN. NVAN HAS NO DEFAULT VALUE.	PTUNU221
r, c	1.14	=	LINE PUINT FLAG AND INDEX, AF (LINERAGO), LINE PUINTS	PTUNO
с С			THE NUT THE ANDER OF THE MACTARIE CON HERCE FRAT	FTUNU223
			LIN ID INE INVEX OF THE VARIABLE FOR WHICH LINIT	PTUNU2,4
r r			TUINTO ANE IN DE DOUDATE LIA MERNULIO IN MERNE I TA MUST CATTERY A LE ITA LE MUAD	PTCN0220
ř	TT	-	TARGET POINT ELAR AND INDEY. LE (IT CO A), TARGET ROINTS	PTCNOCCO
ř	3.1	-	APE NOT TO BE LOOKED FOD ATHEDUTCE, THE HEED CHONED ST	PTCN0.22
ř			IT TO THE INDEX OF THE UARTARIE FOR WHICH TARGET	PTCN0220
č			UAL HES XIT ARE DESTRED. IT HAS THE DEFAULT UAL HE TERD.	ETCN0230
č			IT MAY RE RESET BY THE USED AT ANY TIME DURING A FUN.	PTCN0231
č			IT HIST SATISFY ALF. IT. IF. NUAR.	ETCNOTTO
č	XIT	=	THE VALUE OF THE TARGET DECTOR COMPONENT SOUGHT, LE LL.NE.O.	PTCN0233
č			TARGET POINTS XR SATISFY XR(IT)=XIT, XIT HAS NO DEFAULT.	PTCN0234
č	KSTEP	#	THE NUMBER OF CONTINUATION STEPS TAKEN. THIS DOES NOT	PTCN0235
Ĉ			INCLUDE FAILURES, TARGET POINTS OR LINIT POINTS. THE PROGRAM	FTCN0236
Ĉ			INCREMENTS KSTEP EACH TIME A NEW POINT XF IS COMPUTED.	PTCN0237
Ċ			ON THE FIRST CALL TO PITCON FOR A PARTICULAR PROBLEM, THE	PTC/10238
C			USER SHOULD SET KSTEP TO O OR -1. IF KSTEP=-1, THE PROGRAM	FTCN0239
С			WILL CHECK THE ACCURACY OF THE STARTING POINT XR, AND IF	FTCN0240
С			NECESSARY, ATTEMPT TO CORRECT IT USING NEWTON'S METHOD.	PTCN0241
С			IF KSTEP=0, THE PROGRAM PERFORMS NO CHECK ON THE STARTING	PTCN0242
С			POINT, AND PROCEEDS TO THE CONTINUATION LOOP. IF THE USER	PTCN0243
C			WISHES TO RUN A DIFFERENT PROBLEM THEN A CALL TO PITCON	PTCN0244
C			WITH KSTEP=-1 OR O WILL RESET THE CODE, DESTROYING THE	PTCN0245
C			INFORMATION FROM THE PREVIOUS RUN. KSTEP DEFAULTS TO -1.	PTCN0246
C	IPC	=	THE COMPONENT OF XC TO BE USED AS CONTINUATION PARAMETER.	PTCN0247
Ç			ON THE FIRST CALL ONLY, THE USER DUGHT TO SET IPC OR ALLOW	PTCN0248
Č			THE DEFAULT VALUE IPC=NVAR, AFTER THE FIRST CALL, THE	FTCN0249
Č			DETERMINATION OF IPC IS DONE BY THE PROGRAM USING INFORMATION	IFTENC250
5			ABUUI THE LANGENT VECTOR AT XC.	P10N0251
Č.	н	=	SUBGESTED STARTING STEP SIZE ALUNG THE TANDENT TO THE CURVE.	FIUN0252
с С			IF NEWLOW THE INITIAL CALLS IN DEFAULTS TO (MERATERIN)//	PTUNU203
č			AF A AS NEGATIVE UN THE FIRST CALLT THE MINUS SIGN	PICRU204
с С			ID ADDUKDEN AT DIK AND INDICATED (MAT THE DIRECTION OF THE	FILNU233
ř			AFTER THE FIRST STER, STEREITE IS CONTROLLED BY THE GROUPAN	PTCN0200
с r			HEFTER THE FIRDE DIEFT DIEFDIAE TO LURINULLED DI THE ERUDRADI.	FILRU237
ř			OUEDNOTTEN DY UTANCE, THE CTERCITE HEED IN REACHING THE	PTCN0220
ř			VYENWALITEN DI HIMMUFF ING STEFSIZE USED IN ACHUNING ING. Neu dotat	PTCNO257
ř	TRET	=	A DETINAL FLAG TO INTRATE EPENDS OF THE TYPE OF FOINT	PTCNC241
ř		-	PETIDNER IN YO, NONNEGATINE UNHIES OF TOTAL OF TOTAL	ETCN0247
ř			NORMED IN ART NORMEDITIES OF THE INDIANE THAT	PTENO247
ř		•	COME COORD OF RECEIPTING TO THE AND AND THE THE	STENO243
ř			JUNE ERVER OR DIFFILOUSI AND SEEN ENVIRONMENTS AND	PTCN0245
ř			ATTEMPT TO COMPLETE A LIMIT OF TAPAET POINT FAILED. THESE	PTCN0244
č			AN TERFT TO CONFERE FLATTA ON THOUSE FOR AND THE USES AND	PTCN0200
ř			NOT MODIFY ANY UARIARIES REFORE PROFEDING.	PTCN0248
č			VALUES OF IRET OF -5 AND -6 REFER TO BANGEROUS SITUATIONS	PTCNOCA9
č			THAT HAY BE CORRECTABLE.	PTCH0270
č			VALUES OF TRET FROM -7 TO -10 ARE SERIOUS. FATAL FRANKS.	PTCN0271
č			THE USER SHOULD HALT THE PROGRAM AND EXAMINE HIS INPUT	PTCN0272
č			AND THE INTERIM RESULTS.	PTCN0273
Ĉ			IRET SHOULD BE ZERO ON THE FIRST CALL FOR A PROBLEM.	FTCN0274
Ĉ			THE SPECIFIC VALUES OF IRET AND THEIR MEANINGS ARE:	PTCN0275
Ĉ				PTCN0275
Ċ			IRET=2: NORMAL RETURN WITH LIMIT POINT IN XR AND TANGENT	PTCN0277
С			AT XR CONTAINED IN TL.	PTCN0228
C				FTCN0279
E.			TRFT≠1: NORMAL RETURN WITH TARGET POINT IN XR.	PTENO290

FTCN0281 IRET=0: NORMAL RETURN WITH NEW CONTINUATION FOINT IN XR. PTCN0282 PTCN0283 IRFT=-1: AN ERROR OCCURRED DURING COMPUTATION OF LIMIT POINTPTCN0284 PTCN0285 CORECT CALLED FOR TARGET POINT CALCULATION FAILED PTEN0286 IRET=-2: AFTER KNMAX ITERATIONS. FITCN0287 PTCN0788 SOLVE WAS CALLED BY CORECT FOR TARGET POINT IRET=-3: FTCN0289 CALCULATION, AND FAILED. (MATRIX ELIMINATION FOUNDPICN0290 PTCN0291 7FRO PIVOT). PTCN0292 PTCN0293 UNACCEPTABLE CORRECTOR STEP IN TARGET POINT IRET=-4: CALCULATION. FTCN0294 PTCN0295 PREDICTION STEP HTANCE IS LESS THAN HMIN, PERHAPS BECAUSE OF REPEATED FAILURE OF CORECT, AND CONSEQUENT STEPSIZE REDUCTION. USER MIGHT REDUCE PTCN0296 PTCN0297 IRET=-5: PTCN0298 HMIN, OR SWITCH FROM THOD=1 TO IMOD=0, OR INCREASE PTCN0299 ABSERR AND RELERR. BUT BE AWARE THAT REPEATED STEPSIZE REDUCTIONS MAY INDICATE AN INTRACTABLE PTEN030C PTCN0301 FUNCTION. FTCN0302 PTCN0303 FUNCTION VALUE FNRMXF OF INPUT XR IS TOO LARGE IRET=-6: PTCN0304 AND COULD NOT BE IMPROVED BY CORECT. USER PTCN0305 MIGHT RECOVER BY RELAXING ERROR CONTROLS, IMPROVINGPTCN0306 PTCN0307 STARTING POINT XR, OR CHANGING VALUE OF TPC. PTCN0308 IRET=-7: SOLVE FAILED IN A CALL FROM TANGNT. PTCN0309 PTCN0310 SOLVE FAILED IN A CALL FROM CORECT. PTCN0311 IRET=-8: PTCN0312 IRET=-9: THE TANGENT VECTOR TO AT XO IS ZERO. PTCN0313 PTCN0314 IRET=-10: IMPROPER INPUT, NVAR.LE.1, OR PTCN0315 ISIZE.LT. (NVAR) * (NVAR+5), OR FTCN0316 PTCN0317 PROGRAM HAS BEEN CALLED AGAIN AFTER FATAL ERROR. FTCN0318 IMOD = METHOD FLAG FOR CORRECTOR STEP, SPECIFYING TYPE OF PTCN0319 NEWTON METHOD TO BE USED. PTCN0320 PTCN0321 PTCN0322 IMOD=0: UPDATE JACOBIAN FOR TANGENT CALCULATION, UPDATE JACOBIAN FOR EACH CORRECTOR STEP. PTCN0323 PTCN0324 IMOD=1: UPDATE JACOBIAN FOR TANGENT CALCULATION, EVALUATE JACOBIAN AT FIRST CORRECTOR STEP ONLY. PTCN0325 PTCN0326 PTCN0327 INTEGER VECTOR USER DECLARED TO BE OF SIZE NVAR. USED DURING THE MATRIX FACTORIZATION TO STORE PIVOT TV91 PTCN0328 = PTCN0329 INFORMATION. PTCN0330 HMAX THE MAXIMUM STEP SIZE. IF HMAX.LE.O.O ON INITIAL CALL, PTCN0331 - 2 HMAX DEFAULTS TO SQRT(NVAR). THE MINIMUM STEP SIZE, IF HMIN,LE,SQRT(EPMACH) ON INITIAL CALL, HMIN DEFAULTS TO SQRT(EPMACH), WHERE EPMACH IS THE PTCN033 PTCN0333 HHIN -FTCN0334 MACHINE PRECISION CONSTANT. PTCN0335 HFACT = LIMIT ON STEPSIZE CHANGE. HSECLC IS THE SECANT STEPSIZE PICN0335 OF THE LAST STEP, AND HTANCF IS THE STEPSIZE TO BE USED PTCN0337 IN OBTAINING THE PREDICTED POINT. THE FOLLOWING RELATIONSHIPFTCN0338 HUST BE SATISFIED: (HSECLC/HFACT).LE.HTANCF.LE.(HSECLC#HFACT)FTCN0339 IF THE CORRECTOR STEP FAILS, THEN HFACT IS ALSO USED TO FTCN0340 REDUCE THE PREDICTOR STEP HTANCF TO (HTANCF/HFACT) PTCN0341 IF HFACT.LE.1.0 ON INITIAL CALL, HFACT DEFAULTS TO 3.0. PTCN0342 ABSERR= ABSOLUTE ERROR CONTROL. IF ABSERR=0.0 ON INITIAL CALL PTCN0343 ABSOLUTE ERROR CONTROL. AF HEDLINGTON ON ANALYSIC SHARE ARSERR DEFAULTS TO SORT(EPMACH) RELATIVE ERROR CONTROL. IF RELERR=0.0 ON INITIAL CALL RELERR DEFAULTS TO SORT(EPMACH) USER DECLARED VECTOR OF SIZE ISIZE=NVAR*(NVAR+5). FTCN0344 RELERR= FTCN0345 PTCN0346 RWORK = PTCN0347 RWORK STORES FIVE VECTORS AND AN ARRAY IN THE ORDER (XR;XC;XF;TL;TC;FPRYM). THEIR BEGINNING LOCATIONS ARE IXR=1; IXC=NVAR+1; IXF=2*NVAR+1; ITL=3*NVAR+1; ETCN0348 PTCN0349 PTCN0350 ITC=4*NUAR+1, IFP=5*NUAR+1. FPRYM IS AN ARRAY OF PTCN0351

C

CC

CCCCCCC

SIZE NVAR X NVAR. THE MEANINGS OF THESE COMPONENTS OF PTCN0352 RWORK ARE DESCRIBED BELOW. THE USER SHOULD SET A VALUE TO XEPTCN0353 ON FIRST CALL, BUT NO OTHER PORTIONS OF RWORK SHOULD BE PTCN0354 SET. AFTER THE FIRST CALL FOR A PROBLEM, NO ENTRIES OF RWORNFTCN0355 SHOULD RE ALTERED. FTCN035 9 SHOULD BE HLICKED, A USER SUPPLIED STARTING POINT, WHICH HAY BE IMPROVED BY THE PROGRAM IF KSTEP=-1. ON NORMAL RETURN FROM PITCON, XR WILL HOLD THE MOST RECENTLY FOUND POINT, WHETHER A CONTINUATION POINT, TARGET POINT, OR LIMIT POINT. (XR) FTCN035 PTCN0358 PTCN0357 PTCN0360 = THE PREVIOUS CONTINUATION POINT. = THE CURRENT CONTINUATION POINT. (XC) FTCN0361 (XF) PTCN036 PREVIOUS VALUE OF TANGENT VECTOR. (TE)= NOTE THAT THIS CORRESPONDSPICN0363 (1L) = PREVIOUS VALUE OF TANGENT VECTOR. NOTE THAT THIS CORRESPOND TO A POINT XL WHICH HAS BEEN DISCARDED. ON A LIHIT POINT RETURN, TL WILL CONTAIN INSTEAD THE TANGENT AT THE LIHIT POINT. ON A TARGET POINT RETURN, TL WILL HAVE BEEN OVERWRITTEN BY THE FUNCTION VALUE AT THE TARGET POINT. (TC) = THE TANGENT VECTOR AT THE PREVIOUS CONTINUATION POINT. (FPRYN)= MATRIX STORAGE AREA FOR SETTING UP AND SOLVING THE LINEAR SYSTEMS INVOLVING DEA(X, IP). ISIZE = USER SET DIMENSION FOR UPCTOR PURCH. PTCN0364 PTCN0365 PTCN0366 č č FTCN0367 č FTCN0348 FTCN0369 C PTEN0370 ISIZE = USER SET DIMENSION FOR VECTOR RWORK, WHICH MUST BE AT LEAST OF SIZE NVAR*(NVAR+5). C PTCN0371 Ĉ PTCN0372 PTCN0373 Ċ PTEN0374 NOMENCLATURE FOR STEP DEPENDENT VARIABLES C C C C FTCN0375 PTCN0376 PTCN0377 THE PROGRAM ACCUMULATES INFORMATION THAT IS ASSOCIATED WITH Ċ PTCN0379 THEM. IN INTERPRETING THE CODE OR ITS OUTPUT, IT IS ADDE BETWEEN THEM. IN INTERPRETING THE CODE OR ITS OUTPUT, IT IS INFORTANT TO KNOW WHERE SUCH QUANTITIES APPLY. THE FOLLOWING DESCRIPTION OF SOME OF THE VARIABLES IS VALID ONLY UPON A NORMAL RETURN WITH С PTCN0379 PTCN0380 Ĉ PTCN0381 Ĉ PTCN0382 A CONTINUATION POINT. THE POINTS 'XLL' AND 'XL' WILL HAVE BEEN DISCARDED BY THE PROGRAM. BUT SOME QUANTITIES ASSOCIATED WITH THEM STILL SURVIVE. Ç PTCN0383 C PTCN0384 PTCN0385 PTCN03RS Ċ PUANTITIES ASSOCIATED WITH STEP FROM 'XLL' TO 'XL': PTCN0387 C PTCN0388 HSECLL = SIZE OF SECANT FROM 'XLL' TO 'XL', EUCLIDEAN NORM(XLL-XL) PTCN0389 PTCN0390 C QUANTITIES ASSOCIATED WITH THE POINT 'XL': FTCN0391 PTCN0392 C C C C C C C C THE LOCATION OF THE FIRST OR SECOND LARGEST COMPONENT OF THE TANGENT VECTOR AT 'XL'.
VALUE OF LIM-TH COMPONENT OF TANGENT VECTOR AT 'XL'. IPI. PTCN0393 PTCN0394 TILLIM = PTCN0395 č TANGENT VECTOR AT 'XL', ALTHOUGH LIMIT OR TARGET POINT CALCULATIONS COULD HAVE OVERWRITTEN THIS VECTOR. PTCN0396 PTCN0397 Č PTCN0398 Ĉ C C QUANTITIES ASSOCIATED WITH INTERVAL FROM 'XL' TO XC: FTCN0399 FTCN0400 ALPHLC = THE ANGLE BETWEEN THE TANGENTS AT 'XL' AND XC. FTCN0401 CURVLC = ESTIMATED CURVATURE BETWEEN 'XL' AND XC. č PTCN0402 HSECLC = SIZE OF SECANT BETWEEN 'XL' AND XC, EUCLIDEAN NORM(XL-XC) FTCN0403 PTCN0404 č QUANTITIES ASSOCIATED WITH THE POINT XC: FTCN0405 Ĉ PTCN0406 = BINARY MANTISSA OF DETERMINANT OF DFA(XC, IPL), DIVIDED DETA PTCN0407 BY IPL-TH COMPONENT OF TANGENT AT XC. = SIGN OF DETA, DETERMINES SENSE OF CONTINUATION. = BINARY EXPONENT OF DETERMINANT OF DFA(XC, IPL). = LOCATION_OF FIRST OR SECOND LARGEST COMPONENT OF TANGENT PTCN0408 C C C C DIR PTCN0409 IEXP PTCN0410 Ć IPC PTCN0411 č VECTOR AT XC. TANGENT VECTOR AT XC. PTCN0412 TC -PTCN0413 # VALUE OF LIM-TH COMPONENT OF TANGENT AT XC. # VALUE OF TC(IPC) č TCLIN PTEN0414 TCIPC PTCN0415 PTCN0416 PTCN0417 C QUANTITIES ASSOCIATED WITH THE INTERVAL FROM XC TO XF: C PTCN0418 CURVEF = ESTIMATED CURVATURE BETWEEN XC AND XF. HTANEF = STEPSIZE USED ALONG TANGENT TO GET PREDICTED POINT PTCN0419 PTCN0420

		DTCNOADS
2	WHICH WAS CURRECTED TO SOLUTION POINT XF.	PTCN0421
Ç		PTUNU412
C	QUANTITIES ASSOCIATED WITH THE PUINT XF:	P1080423
C	THE THE THE THE THE TELL PROVE THE TELL PROVE THE TELL	PILN0424
C	CORDIS = SIZE OF THE TOTAL CORRECTION FROM FREDITIED FOUNT	FILR042.J
0	X=XC+HTANCF*TC TO CORRECTED POINT XF.	PIUNU420
C	NOTE THAT THIS HAS BEEN MUDIFIED TO AN OPTIMAL' VALUE.	P1UN0427
C	CURVXF = A PREDICTED VALUE OF THE CURVATURE AT XF.	F10N0428
0	FNRMXF = MAXIMUM NORM OF FUNCTION VALUE AT XF.	P11.NU4/27
C	FPRYN = DFA(XF, IPC) HAS ACTUALLY BEEN LAST EVALUATED AT THE	PTCN0430
C	PENULTIMATE CORRECTOR ITERATE (IF IMOD.NE.1). IT WILL BE	PTUN0431
С	EVALUATED AT XF AS SOON AS THE NEXT LOOP REGINS AND THE	FICN0432
Ĉ	TANGENT IS NEEDED.	FIUNU432
Ç.	XSTEP = SIZE OF THE LAST CORRECTOR STEP TAKEN IN CONVERSING TO XF-	FTCN0434
C		FILNU450
Ç		PTCN0430
Č	SUBROUTINES IN THIS PACKAGE	PTCN0432
Č		DTCNA470
C		F 10499407
Č.	PITCON(NVAR,LIN,IT,XIT,KSTEP,IPC,H,IRE),IMOD,IPVT,	PTCN0440
Č.	HMAX+HMIN+HFACT+ABSERR+RELERR+RWORK+ISIZE)	PTCN0441
č		PTCN0442
Č	DRIVING ROUTINE OF CONTINUATION CODE. INITIALIZES INFORMATION,	PTCN0443
Ç	DETERMINES WHETHER LIMIT, TARGET OR CONTINUATION POINT WILL	PTCN0444
Ç	BE SOUGHT THIS STEP, COMPUTES STEPLENGTHS, CONTROLS CORRECTOR	PTCN0445
Č.	PROCESS, AND HANDLES ERROR RETURNS,	PTCN0446
C		PTCN0447
č	CORECT(NVAR,X, HOLD, WORK, IERR, INOD, FPRYM, IPVT, ABSERR, RELERR,	PTCN0448
Ľ	XS(EP)NEUN)FNKH)	PTCN0449
č	HORE & FORM OF MERICAN A VETUER TO COUNT THE ANALYSINES MAN AND A	PTCN0450
Ľ,	USES A FURH UP NEWTON'S RETHOD TO SULVE THE AUGMENTED NONLINEAR	PTCN0451
Č,	SISTER FACX)=0 WITH AUGRENTING EUGATION (INULU)=8, (HAT 15, X(INULU)	PICN0452
	IS HELD FIXED DURING THE CURRECTION PROCESS.	PIUN0453
ř	TANONT/MIAD. VC. TOC. TO TOET TOALL COONS. TOUT MEDAL DETA TOVON	PIUNU434
	THIRDRY (NYHRYACHIECHICHICHIARET)	PILINU433
č	ADDITES ALCODITIN DECEDTED ADDIE TO SOLUE DEALYS, TOLATS-ELMIADA	P10.N0406
ř	AND THEN NOVALLATE TANGENT HEAVE AN OBSECTE STALLATIF. / ALD THE TANGENT	FILNV43/
ř	THE AND ALE	PTCN0450
ř		PTCN0437
č	ROOT(A+FA+R+FR+C+FC+KOUNT+TFLAG)	PTCN0441
č		PTCN0462
č	ROOT FINDER USED TO LOCATE LINIT POINT. THIS ROUTINE IS A MODIFIED	PTCN0463
Ē	VERSION OF THE FORTRAN FUNCTION ZERO GIVEN IN THE BOOK:	PTCN0464
č	ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES	PTCN0465
Ĉ.	BY RICHARD P BRENT, PRENTICE HALL, 1973.	PTCN0466
Ĉ		PTCN0467
Ĉ.	SOLVE(NVAR,X,Y, IP, DETA, IEXP, IERR, ICALL, IMOD, FPRYM, IPVT)	PTCN0468
Ċ		PTCN0469
C	SETS UP AND SOLVES THE SYSTEM DFA(X+IP)*Y(OUTPUT)=Y(INPUT)	PTCN0470
C	WHERE DFA(X, IP) IS THE JACOBIAN OF FA AT X,	PTCN0471
C	AND Y IS A RIGHT HAND SIDE SUPPLIED BY THE CALLING ROUTINE.	PTCN0472
C		PTCN0473
Č.	**NOTE** SUBROUTINE SOLVE USES FULL MATRIX STORAGE TO SOLVE THE	PTCN0474
С	SYSTEM. THE USER MAY WISH TO REPLACE THIS ROUTINE WITH ONE MORE	PTCN0475
Č.	SUITED TO HIS PROBLEM.	PTCN0476
Č		FTCN0477
Č		FTCN0478
Č.	USER SUPPLIED SUBRUUTINES	PTUN0479
2		FILN0480
L C		FICN048]
č		FILRUADZ
ř	CHANNER THE MIAD & COMPONENT CHARTEN AN AND AN AND COMPONENT	FILRVARS
ř	COMPUNED THE NUMBER CONFUNENT FUNCTION FACTOR A MAIN WAR CONFUNENT UPFENDED THE NOMENTAL CONFUNENTIAL	FILRU484
ř	- VERTERA THAN FURNITAR REPORTED THE RURLINERS DIDIED THE MUMBRITARS - Constinue to Mannier by the continuation darkage	10000000
ř	ENDMIANT AN DEMARCH RI FOR GURIARUMIAUN FMGRMUCS	FTCN0400
ř	FRETHE (NUAR, Y. FREYN)	PTCNA200
č	+ T N & FM () 17 7 M 7 M 7 M 7 M 7 M 11 2	PTCNOARO
č	FUALIJATES THE NUAR-1 BY NUAR JACOBTAN MATRIX FPRYM (X)	PTENGA90
	A COMPANY AND COMPANY A 201 OFFICE AND ADDREADED TOTAL AND A COMPANY AND A COMPANY ADDREADED AND A COMPANY ADDR	· · · · · · · · · · · · · · · · · · ·

•

Contraction of the second s

AT X AND STORES IT IN THE NUAR BY NUAR ARRAY FERYM. PTCN0491 THE LAST ROW OF FPRYM (FOR THE AUGMENTING EQUATION) IS INSERTED PTCN049? PTCN0493 BY THE ROUTINE SOLVE. Ĉ PTCN0494 £ **FTCN0495** PTCN0496 PTCN0497 C C LINPAK ROUTINES USED C PTCN0498 Ĉ FTCN0499 LINPAK REFERENCE 0000 PTCN0500 LINPACK USER'S GUIDE PTCN0501 J J DONGARRA, J R BUNCH, C B MOLER AND G W STEWART, SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS, PTCN0502 Ĉ PTCN0503 PHILADELPHIA, 1979. PTCN0504 PTCN0505 C PTCN0506 C ISAMAX (N, SX, INCX) PTCN0507 Č INTEGER FUNCTION RETURNS THE POSITION OF LARGEST ELEMENT OF SX PTCN0508 С PTCN0509 C SAXPY(N, SA, SX, INCX, SY, INCY) PTCN0510 C SETS VECTOR $SY(I) = SA \pm SX(I) + SY(I)$ PTCN0511 C PTCN0512 SCOPY(N, SX, INCX, SY, INCY) PTCN0513 č SETS SY(I)=SX(I) PTCN0514 Ĉ PTCN0515 Ē SDOT (N+ SX+ INCX+ SY+ INCY) PTCN0516 SBOT = SUM(I=1 TO N) SX(I) * SY(I)PTCN0517 C PTCN0518 C SNRM2(N+SX+JNCX) PTCN0519 Ċ SNRM2 = EUCLIDEAN NORM OF SX(I) PTCN0520 C PTCN0521 Č ****NOTE**** SNRM2 HAS HACHINE DEPENDENT CUTOFF CONSTANTS PTCN0522 С PTCN0523 C SSCAL (N, SA, SX, INCX) PTCN0524 SETS SX(I)=SA*SX(I) Č PTCN0525 C PTCN0526 С SGEFA(A,LDA,N, JPVT, JNFO) PTCN0527 ACTORS MATRIX A WHOSE LEADING DIMENSION WAS DECLARED AS LDA AND WHOSE ACTUAL USED DIMENSION IS N, SETS UP PIVOT INFORMATION IN VECTOR IPVT AND WARNS OF ZERO PIVOTS. C PTCN0528 PTCN0529 C PTCN0530 PTCN0531 SGESL(A,LDA,N,IPVT,B,JOB) ACCEPTS OUTPUT OF SGEFA, AND A RIGHT HAND SIDE B, AND SOLVES SYSTEM A*X=R, RETURNING X IN B. FOR MODIFIED NEWTON'S METHOD, ONCE MATRIX IS FACTORED BY SGEFA, ONLY SGESL IS CALLED FOR SUCCESSIVE PTCN0532 PTCN0533 C PTCN0534 Ĉ PTCN0535 č PTCN0536 PTCN0537 RIGHT HAND SIDES PTCN0538 C PTCN0539 LABELED COMMON BLOCKS C PTCN0540 PTCN0541 Ĉ PTCN0542 /COUNT1/ COUNTS NUMBER OF CALLS FROM ... TO ... AS FOLLOWS: PTCN0543 = CORECT TO SOLVE = TANGNT TO SOLVE = PITCON TO CORECT FOR IMPROVED STARTING POINT = PITCON TO CORECT FOR CONTINUATION POINT = PITCON TO CORECT FOR CONTINUATION POINT = PITCON TO CORECT FOR CONTINUATION POINT C C C C C C ICRSI. PTCN0544 PTCN0545 I TNSI. PTCN0546 NSTCR PTCN0547 NCNCR = PITCON TO CORECT FOR TARGET POINTS = PITCON TO CORECT FOR LIHIT POINT Ĉ NTRCR PTCN0548 NL.MCR PTCN0549 C NLMRT = PITCON TO ROOT FOR LIMIT POINT PTCN0550 PTCN0551 NOTE THAT NSTCR, NCNCR, NTRCR, NLHCR AND NLMRT COUNT THE NUMBER OF ITERATIVE STEPS (NEWTON OR ROOTFINDING) AND NOT JUST THE NUMBER OF SUBROUTINE CALLS. PTCN0552 PTCN0553 CCC PTCN0554 C PTCN0555 /COUNT2/ KEEPS PERFORMANCE AND WORK STATISTICS IFEVAL = NUMBER OF CALLS TO FCTN IPEVAL = NUMBER OF CALLS TO FPRIME ISOLVE = NUMBER OF CALLS TO SOLVE PTCN0556 PTCN0557 C PTCN0558 Ĉ PTCN0559 = NUMBER OF STEPSIZE REDUCTIONS MADE BEFORE PREDICTOR C NRFI PTCN0560

C	POINT CONVERGED TO THE NEW CONTINUATION POINT.	PTCN0561
C	NRDSUM = TOTAL NUMBER OF STEPSIZE REDUCTIONS	PTCN0562
С	KN = NUMBER OF CORRECTOR ITERATION STEPS TAKEN IN MOST RECENT	PTCN0563
C	CALL TO CORFET.	PTCN0564
Ĉ	KNSUM = TOTAL NUMBER OF CORRECTOR ITERATION STEPS.	PTCN0565
Č.		PTCN0566
č		PTCN0547
ř		PTCNAFIG
ř	INALLE - USEN NUCESSINCE DUTEDI ANNANTUNI IURITESA, NA ANTONI DETATEN BY RITCAN	DICNOSAD
ř	INFJIE-VY RU DUTUL FRINTED DI FILLUN. Iuditent, Eddo Weckere Dointed by Diteon	PTCN0570
2	INALLET / CARUN REDOMBED FRANCES AL FALLAND INSTEED CERTAIN ONTOIL ULL DE ODINTED BY DITCON	PTCN03/0
	IWRITE=29 CERTAIN UNIPUT WILL BE PRINTED BY PATLON.	PTUNU5/1
Ŭ,		PTCN0572
Ľ.	PUINT CUNTAINS DATA ABOUT THE SOLUTION CORVE	PTUN0573
Ū.	DETA = BINART MANTISSA OF THE DETERMINANT OF THE AUGMENTED JACUBIA	NPTCN05/4
Ç	1EXP = BINARY EXPONENT OF THE DETERMINANT OF THE AUGMENTED JACOBIA	NPTCN0575
Ğ	CURVEF = ESTIMATED CURVATURE BETWEEN XC AND XF.	PTCN0576
Ç	CORDIS = NORM OF THE CORRECTOR STEP FROM PREDICTED POINT TO CORRECTE	DETCN0577
2	POINT, USING MAXIMUM ABSOLUTE VALUE AS THE NORM	PTCN0578
C	THIS QUANTITY IS MODIFIED TO AN "OPTIMAL" VALUE.	PTCN0579
Ç	<u>ALPHLC = ANGLE BETWEEN OLD AND NEW TANGENTS TL AND TC</u>	PTCN0580
C	HSECLC = EUCLIDEAN NORM OF SECANT BETWEEN XL AND XC+	PTCN0581
Ç	FNRMXF = MAXIMUM NORM OF FUNCTION VALUE AT NEW CONTINUATION POINT.	PTCN0582
C		PTCN0583
C	/TOL/	PTCN0584
C	EPHACH= SHALLEST NUMBER SUCH THAT 1.04EPHACH.GT.EPHACH	PTCN0585
С	•5*BETA**(1-TAU) FOR ROUNDED, TAU-DIGIT ARITHMETIC	PTCN0586
0	BASE BETA. TWICE THIS VALUE FOR TRUNCATED ARITHMETIC.	PTCN0587
С	THIS IS THE RELATIVE MACHINE PRECISION.	PTCN0588
Č	EPMACH=2**(-27) FOR DEC-10.	PTCN0589
Ċ	EPSATE= 8×EPMACH	PTCN0590
Ĉ	EPSORT = SQUARE ROOT OF EPHACH	PTCN0591
č		PTCN0592
č		PTCN0593
Ē.	PROGRAMMING NOTES	PTCN0594
č		PTCN0595
ē.		PTCNOSOL
č	THE LISER HUST -	PTCN0597
č		PTCN0598
č	1. WRITE SURROUTINES	PTCN0599
č	SUPPLY & CALLING PROGRAM. AND THE THE RELITINES FORM AND FRAME	PTENOADO
č	AS RESCRIBED ABOVE.	PTCN0601
č		PTCN0602
č	2. SET STORAGE AREAS	PTCN0603
č	NECLARE & REAL VECTOR RHORK OF SITE ISITE. ISITE. GE. NUARX(NUAR+5)	PTCN0604
č	AND AN INTEGER VECTOR IPUT OF SIZE NUAR.	PTCN0605
č		PTCN0404
ē	3. PASS FERTAIN NON-DEFAIL TARLE VALUES	PTEN0407
ř	DACE NUAD OPENTED THAN TENT TELT (ALAPT (NUAPTS)	PTCNOADR
ř	AND SET DETEN. KETEDENI DE KETEDEN AN FIRST CALL	PTCNOA09
ř		PTENOAIO
ř	F CPC PF FEED F NORMEDLEED	PTCM0411
ř		PTCNAL12
ř	THE USER SHORE	PTENOLIZ
ř	1. STORE & STATING DUINT YO IN THE FIRST NUAR HORATIONS OF PUNCH	PTCNOLIA
ř	AF BIDLE B DIMILING FURNI AN AN INC FARDI NYAN LIRAFIADAD OF DWONN	PTCNALIS
ž	PECANE UNLING FILADA CHEN THE ODDE MAY DE INADA E TO DEDUCE OUT	FILMUOLU
ž	IF SUCH A VALUE IS NOT GIVENT THE LOUE MAY BE UNABLE TO PRODUCE ONE.	PTCN0616
ř	2 CAREFULLY MONTTON THE HALVE OF THET OF THAT ANY OFFICE FROM	PTUN0617
ž	TE CHICHT DECORTANT CALL TO MADE TO TAKE TO THE ANT SER COUS ERROR	PTUN0618
ž	AB CHOOME BEFUKE MAUTHER LALL IS MADE IU	PTUN0619
ř	7. CHORSE A VALUE OF THOSE FOR THE TYPE OF CONSECTOR PROFESSION	FTUN0620
ř	DE LECTA DE VICUE VE ANUS FUR LICE TEE DE CURRELIUR PRIJESS TO	PTUN0621
ž	RE UGENO	FTCN0622
5		PTCN0623
5		PTCN0624
ž	ITE USER THE T	PTCN0625
Ľ,		PTCN0626
U.	A TURITUR THE PASSING OF BIFURCATION POINTS BY SAVING THE OLD	PTCN0627
Ľ,	VALUE UP DETA ANU CUMPARING IT TO THE CURRENT VALUE, IF THERE	PTCN0628
Č,	IS A UNAMOR IN STANT THEN A BIFURCATION POINT HAS BEEN PASSED.	PTCN0629
• •		PTEN0630

and the second second

) | |

Č 2. ACCESS THE COMMON BLOCKS /COUNTI/ AND /COUNTI/ TO KEEP TRACK PTCN0631 PTCN0632 OF THE ANQUNT OF WORK DONE. Ç PTCN0633 MONITOR THE CONNON BLOCK /POINT/ FOR INFORMATION ABOUT THE Ĉ 3. PTCN0634 Ĉ PTCN0635 SOLUTION CURVE. C PTCN0636 Ē AT ANY TIME, RESET THE CODE BY PASSING IN KSTEP=-1 OR KSTEP=0. PTCN0637 THIS ALLOWS THE USER TO CHANGE STEPSIZE, DIRECTION OF CONTINUATION, ERROR CONTROLS, OR OTHER PARAMETERS. IT ALSO ENABLES THE USER TO RUN UNRELATED PROBLEMS OF DIFFERENT SIZES OR ERROR CONTROLS C PTCN0638 č PTCN0639 PTCN0640 C Ĉ DURING A SINGLE PROGRAM EXECUTION. PTCN0641 C PTCN9642 č PTCN0643 č PTCN0644 THIS SUBROUTINE IS CALLED BY USER MAIN PROGRAM PTCN0645 PTCN0646 PTCN0647 CORECT ROOT PTCN0648 TANGNT PTCN0649 FORTRAN ABS PTCN0650 FORTRAN ACOS PTCN0651 FORTRAN ALOG PTCN0652 PTCN0653 FORTRAN AHIN1 PTCN0654 FORTRAN PTCN0655 DPL.E. PTCN0656 FORTRAN FLOAT FORTRAN SIGN PTCN0657 FORTRAN SIN PTCN0658 FORTRAN PTCN0659 SNGL FORTRAN SORT PTCN0660 LINPAK ISAMAX PTCN0661 PTCN0662 LINPAK SAXPY LINPAK SCOPY PTCN0663 LINPAK SNRH2 PTCN0664 LINPAK SSCAL PTCN0665 Ċ PTCN0666 PTCN0668 C PTCN0669 INTEGER IPUT (NVAR) REAL RWORK(ISIZE) REAL WRGE(8), ACOF(12) PTCN0670 PTCN0671 DOUBLE PRECISION DILIPC,DICIPC,DAD.HUS,COSALF COMMON /COUNT1/ ICRSL,IINSL,NSTCR,NCNCR,NTRCR,NLMCR,NLMRT COMMON /COUNT2/ IFEVAL,IPEVAL,ISOLVE,NRED,NRDSUM,KN+KNSUM COMMON /OUTPUT/ IWRITE PTCN067 PTCN0673 PTCN0674 PTCN0675 TETA, TEXP, CURVCF, CORDIS, ALPHLC, HSECLC, FNRMXF EPMACH, EPSATE, EPSQRT PTCN0676 PTCN0677 COMMON /POINT/ COMMON /TOL/ DATA IDONE /0/ /0/ PTCN0678 /0.1/ PTCN0679 DATA TENM1 DATA TENH2 /0.01/ PTCN0680 DATA TENMS /0.001/ PTCN0681 DATA WRGE PTCN0682 1 .8735115E+00, .1531947E+00, .3191815E-01, .3339946E-10, PTCN0683 .4677788E+00, .6970123E-03, .1980863E-05, .1122789E-08/ PTCN0684 PTCN0685 DATA ACOF .9043128E+00,-.7075675E+00,-.4667383E+01,-.3677482E+01, .8516099E+00,-.1953119E+00,-.4830636E+01,-.9770528E+00, PTCN0686 PTCN0687 3 .1040061E+01, .3793395E-01, .1042177E+01, .4450706E-01/ PTCN0688 PTCN0689 C PTCN0691 PREPARATIONS PTCN069 C ON FIRST CALL FOR THIS PROBLEM, INITIALIZE COUNTERS AND VARIABLES, CHECK USER INFORMATION AND SET DEFAULTS, AND IF (KSTEP.EQ.-1), CHECK NORM OF F(XR) AND CORRECT XR IF NECESSARY. ON EACH CALL, IF INPUT IRET HAS NONFATAL VALUE, RESET IRET SO THAT CONTINUATION LOOP PICKS UP WHERE IT WAS HALTED. PTCN0693 FTCN0694 C PTCN0695 PTCN0696 PTCN0697 C PTCN0698 PTCN0700 C

TERR=0 (TRET.EQ.-1) TRET=2 ŤĒ ÎF (IRET.EQ.-2.OR. JRET.EQ.-3.OR. IRET.EQ.-4) IRET#1 1F (IRET,EQ.-S.OR.IRET.EQ.-6) IRET=0 Ĉ. TE CODE WAS CALLED AGAIN AFTER FATAL VALUE OF TRET, THEN RETURN WITH ERROR VALUE IRET=-10. C С JF (JRET.LT.0) G0 T0 440 С č PERFORM ONE-TIME ONLY INITIALIZATIONS č IF (INONE, NE.O) GO TO 10 С SET THE MACHINE DEPENDENT VARIABLE EPMACH, THE SMALLEST NUMBER C С SD THAT (1,0+EPNACH, GT, 1,0) Ĉ Ĉ FOR DEC POP-10 IN SINGLE PRECISION: ć EPNACH#7,4505806E-9 C Ī, FOR IBN 360 OR 370 IN SHORT (SINGLE) PRECISION: С Ĉ EPNACH=9.53674E-7 C C C C FOR COC 6600 OR 7400 IN SINGLE PRECISION: Ĉ EPNACH=7.105427406E-15 C SET EPSATE=3#EPNACH, EPSQRT=SQRT(EPNACH) Ĉ EPSATE=8,0#EPMACH EPSORT=SORT(EPHACH) ALFHIN=2.0#ACOS(1.0-EPHACH) IF (KSTEP.LT, -1.OR.KSTEP.GT.O)KSTEP=-1 KSTEPO=-2 TOONE=1 C C PERFORM INITIALIZATIONS AND CHECKS FOR NEW PROBLEM ONLY 10 IF (KSTEP, GT, 0) GO TO 30 IF (KSTEPO.EQ. -1. AND.KSTEP.EQ.0)GO TO 30 TE (NYAR LE 1) BO TO 440 IF (ISIZE.LT. (NVAR)*(NVAR+5)) GO TO 440 IXR=1 JXR=0 **IXC=IXR+NVAR** JXC=JXR+NVAR TXF=TXC+HVAR JXF=JXC+NVAR TTI.=TXF+NVAR JTL=JXF+NVAR TTC=TTL+NVAR JTC=JTL+NVAR **IFP=ITC+NVAR** JFP=JTC+NVAR RETA=0.0 TCIPC=0.0 CURDIS=0.0 CURVCF=0.0 HSECLL≖0.0 HSECLC=0.0 XIJ0=0.0 ITO=0 NEON=NVAR-1 NRED=0 KNSUM=0 NRDSUM=0 ICRSL=0 ITNSL=0

44

PTCN0701

PTCN0702

PTCN0703

PTCN0704 PTCN0705

PTCN0704

PTCN0707

PTCN0708

PTCN0709

PTCN0710

PTCN0711

PTCN0712

PTCN0713

PTCN0714

PTCN0715

PTCN0716

PTCN0717

PTCN0718

PTCN0719

PTCN0720

PTCN0721 PTCN0722

PTCN0723

PTCN0724 PTCN0725 PTCN0726

PTCN0727

PTCN0728

PTCN0729

PTCN0730

PTCN0731 PTCN0732

PTCN0733 PTCN0734

PTCN0735

PTEN0736 PTEN0737 PTEN0738

PTCN0739 PTCN0740 PTCN0741

PTCN0742

PTCN0743

PTCN0744 PTCN0745 PTCN0746

PTCN0747

PTCN0748 PTCN0749 PTCN0750

PTCN0751

PTCN0752

PTCN0753 PTCN0754

PTCN0755

PTCN0756

PTCN0757

PTCN0758

PTCN0759

PTCN0760

PTCN0761

PTCN0762

PTCN0763

PTCN0764

PTCN0765 PTCN0766

PTCN0767 PTCN0768

PTCN0769 PTCN0770

NSTCR=0 PTCN0771 NCNCR=0 PTCN0772 NTRCR=0 PTCN0773 NL MCR=0 PTCN0774 NI_MRT=0 PTCN0775 IFEVAL =0 PTCN077A ISOLVE=0 PTCN0777 IPEVAL =0 PTCN0778 IF (HMAX.LE.0.0) HMAX=SQRT(FLUAT(NVAR)) IF (HMIN.LE.EPSQRT) HMIN=EPSQRT PTCN0779 PTCN0780 HDEF= . 5* (HMAX+HMIN) PTCN0781 TF (HFACT,LE,1,0) HFACT=3.0 HRED=1.0/HFACT PTCN0782 PTCN0783 (ABSERR,LE,O.O) ABSERR=EPSORT (RELERR.LE.O.O) RELERR=EPSORT (TPC.LE.O.OR, [PC.GT.NVAR) IPC=NVAR (LIM_LT.O.GR.LIM_GT.NVAR) LIM=0 TF TF PTCN0784 PTCN0785 PTCN0786 IF IF PTCN0787 ĪĒ (H.EQ.0.0) H=HDEF PTCN0788 DIR=SIGN(1.0,H) PTCN0789 H=ABS(H) PTCN0790 PTCN0791 IF (KSTEP.LT.O) CHECK NORM OF F(XR) AT STARTING POINT, IF ACCEPTABLE, RETURN (MHEDIATELY WITH KSTEP=0, OTHERWISE APPLY NEWTON'S METHOD, HOLDING VALUE OF č PTCN0792 С PTCN0793 PTCN0794 C C TPC-TH COMPONENT FIXED. PTCN0795 ř PTCN0796 PTCN0797 PTCN0798 PTCN0799 1 NSTCR=NSTCR+KN PTCN0800 PTCN0801 IF NO ACCEPTABLE POINT FOUND, ERROR RETURN PTCN0802 С PTCN0803 1F (IERR.NE.0) GD TO 400 PTCN0804 KSTEPD=-1 PTCN0805 KSTEP=0 PTCN0806 HTANCF=H PTCN0807 GO TO 340 20 TF (KSTEP.EQ.0) CALL SCOPY(NVAR;RWORK(IXR);1;RWORK(IXC);1) IF (KSTEP.ER.0) CALL SCOPY(NVAR;RWORK(IXR);1;RWORK(IXF);1) PTCN0808 PTCN0809 PTCN0810 PTCN0811 C *****PTCN0812 PTCN0813 С 2. TARGET POINT CHECK. IF (JT.NE.O) TARGET POINTS ARE SOUGHT. CHECK TO SEE IF TARGET CUMPONENT IT HAS VALUE XIT LYING BETWEEN XC(IT) AND XF(IT). IF SO, GET LINEARLY INTERPOLATED Ĉ PTCN0814 č PTCN0815 PTCN0816 STARTING POINT, AND USE NEWTON'S HETHOD TO GET TARGET POINT C PTCN0817 PTCN0818 #PTCN0819 С PTCN0820 30 IF(IT)LT.0.0R.IT.GT.NYAR)IT=0 PTCN0821 IF (IT.EP.0) GO TO 40 IF (IRET.EQ.1.AND.XIT.EQ.XIT0.AND.IT.EQ.IT0) 60 TO 40 PTCN0822 PTCN0823 XCIT=RWORK(JXC+IT) PTCN0824 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 PTCN0825 PTCN0826 PTCN0827 PTCN0828 RAT=0.0 PTCN0829 TF (ARS(DEL).GT.EPSORT) RAT=(XIT-XCIT)/NEL CALL SCOPY(NYAR,RWORK(IXF),1,RWORK(IXR),1) CALL SAXPY(NYAR,-1.0,RWORK(IXC),1,RWORK(IXR),1) PTCN0830 PTCN0831 PTCN0832 CALL SSCAL (NVAR, RAT, RWORK(IXR),1) CALL SAXPY(NVAR, 1.0, RWORK(IXC),1, RWORK(IXR),1) PTCN0833 PTCN0834 RWORK(JXR+TT)=XIT PTCN0835 CALL CORECT(NVAR, RWORK(IXR), JT, RWORK(JTL), IERR, JHOD, RWORK(JFP), PTCN0836 (PUT + ABSERR + RELERR + XSTEP + NEON + FNRM) PTCN0837 1 NTRCR=NTRCR+KN PTCN0838 FTCN0839 TT0=IT XTTN=XTT PTCN0840

45

AL 1. 1. 1999 . 14

C C #:	IF (JERR.EQ.O) 60 TO 320 IF (JERR.EQ1) 60 TO 370 IF (JERR.EQ2) 60 TO 360 IF (JERR.EQ3) 60 TO 380	PTCN0841 PTCN0842 PTCN0843 PTCN0844 PTCN0845 #PTCN0846
, ,	3, TANGENT AND LOCAL CONTINUATION PARAMETER CALCULATION, UNLESS THE TANGENT AND LIMIT POINT CALCULATIONS WERE ALREADY PERFORMED (BECAUSE THE LOOP MAS INTERRUPTED FOR A LIMIT POINT), SET UP AND SOLVE THE EQUATION FOR THE TANGENT VECTOR, FORCE THE TANGENT VECTOR TO BE OF UNIT LENGTH, AND FORCE THE IPL-COMPONENT TO HAVE THE SAME SIGN AS THE IPL-TH COMPONENT OF THE PREVIOUS TANGENT VECTOR, OR (ON FIRST STEP) THE SAME SIGN AS THE USER INPUT DIRECTION OIR. SET THE LOCAL PARAMETER IPC TO THE LOCATION OF THE LARGEST COMPONENT OF THE TANGENT VECTOR, UNLESS A LIMIT POINT IN THAT DIRECTION AFPEARS TO RE AFPROACHING AND ANOTHER CHOICE IS AVAILABLE.	PTCN0847 EPTCN0848 PTCN0850 PTCN0850 PTCN0851 PTCN0853 PTCN0854 PTCN08554 PTCN08556 PTCN0855 PTCN0858 PTCN0858
C	· · · · · · · · · · · · · · · · · · ·	PTCN0860
	40 JF (IKET-NE-2) 60 TO 50 TRET=0	PTCN0861 PTCN0862
С	FG TG 160	PTCN0863
č	STORE OLD TANGENT IN TL, COMPUTE NEW TANGENT FOR XC	PTCN0865
L	50 JPL=JPC	PTCN0867
	IF(KSTEP.GT.O)CALL_SCOPY(NVAR;RWORK([TC);1;RWORK([TL);1) ICALL=1	PTCN0868 PTCN0869
	CALL TANGNT (NYAR, RWORK (IXF), IPC, RWORK (ITC), IRET, ICALL, RWORK (IFP),	PTCN0870
	TF (TRET,EQ,-2) GD TO 430	PTCN0871
r	IF (IRET.EP1) GO TO 410	PTCN0873
č	SUBROUTINE TANGNT RETURNED IPC, THE LOCATION OF THE LARGEST COMPONEN	FPTCN0875
С С	OF THE TANGENT VECTOR. THIS WILL BE USED FOR THE LOCAL PARAMETERIZATION OF THE CURVE UNLESS A LIMIT POINT IN IPC SEEMS	PTCN0876 PTCN0877
Č.	TO BE COMING. TO CHECK THIS, WE COMPARE TO POS=TO(IPC) AND THE	PTCN0878
č	THAN 1 OF TCIPC, AND IC(JPC) IS LARGER THAN TL(JPC),	PTCNO880
C C	WHEREAS TC(IPC) IS LESS THAN TL(IPC)+ WE WILL RESET THE LOCAL PARAMETER IPC;=JPC.	PTCN0881 PTCN0882
С		PTCN0883
•	TCIPC=RWORK(JTC+IPC)	PTCN0885
	JPC=\PC IF(ABS(TCJPC),GE.ABS(RWORK(JTL+IPC)))GO_T0_60	PTCN0886 PTCN0887
	1F(TL1PL,E0.0.0)60T0 60	PTCN0888
	JPC=ISAHAX(NVAR,RWORK(ITC),1)	PTCN0890
	TCJPC=RWORK(JTC+JPC) RWORK(JTC+TPC)=TCTPC	PTCN0891 PTCN0892
	IF (ABS(TCJPC), LT, TENN1#ABS(TCJPC)) GQ TQ 60 TE (ABS(TCJPC), LT, ABS(PUDEK(JTL)) GDT() 40	PTCN0893
	IPC=JPC	PTCN0895
	IF (IWRITE,GE,2)WRITE(6,610) 60 TCIPL=RWORK(JTC+IPL)	PTCN0896 PTCN0897
	DTI TPC=DBLE(RWORK(JTL+TPC))	PTCN0898
ç		PTCN0900
C	CONPARE THE SIGN OF TC(TPL) WITH SIGN OF TL(TPL)	PTCN0901 PTCN0902
C	(RUT ON FIRST STEP; COMPARE SIGN OF TC(JPL) WITH USER INPUT DIR).	PTCN0903
č	AND THE SIGN OF DETA.	PTCN0905
C C	THEN RECORD DIRT# SIGN OF DETERMINANT # SIGN(DETA).	PTCN0904 PTCN0907
	STITET AND AN STITETERS (1 A-TITET)	PTCN0908
·.	tr (stan(1,0,tctpl), Eq. stl tpl) go to 70	PTCN0910

ľ

ł

- 6-

	CALL SSCAL (NUAR1.0.RHORK(ITC).)	PTCN0911
	DETA=-DETA	PTCN0912
	TCIPL=-TCIPI.	PTCN0913
	70_TCTPC=RWORK(JTC+TPC)	PTCN0914
	TCJPC=RWORK(JTC+JPC)	PTCN0915
	DICIPC#UBLE(ICIPC)	P1CN091A
	$\begin{array}{cccc} \mathbf{U}(\mathbf{x}=\mathbf{y})\mathbf{G}(\mathbf{x}), \mathbf{U}(\mathbf{x})=\mathbf{H} \\ \mathbf{U}(\mathbf{x}=\mathbf{y})\mathbf{G}(\mathbf{x}), \mathbf{U}(\mathbf{x})=\mathbf{H} \\ \mathbf{U}(\mathbf{x}), $	PICN0717
		PTCN0919
	TČL TN=RŇOŘK (JTC+L TN)	PTCN0920
С		PTCN0921
Ç	COMPUTE ALPHIC, THE ANGLE BETWEEN TANGENT AT XL AND TANGENT AT XC	PTCN0922
C	AND HSECLC, THE EUCLIDEAN NORM OF SECANT FROM XL TO XC.	PTCN0923
С		PTCN0924
	80 (F(KSTEP.LE.O) 60 10 180	P10N0925
	LUSALF = U. 000 DO 90 T-1. MUAD	PIGNUT20
		PTCN0928
		PTCN0929
		PTCN0930
	HSECLC=SNRM2(NVAR, RWORK(IXR), 1)	PTCN0931
	ALPHLC=SNGL(COSALF)	PTCN0932
	IF (ALPHLC.BT.1.0) ALPHLC=1.0	PTCN0933
	TF(ALPHLC+LT+-1,0)ALPHLC=-1,0	PTCN0934
	ALPHLC=ACOS(ALPHLC)	PTCN0935
-	tf(twrite.ge,2)write(6,550)alphlc	PICN0936
Č.		PICN0937
្ពុប្តុគ		01000700 01000070
ř	A. LINTT POINT CHECK. JE (LIN.NE.O) CHECK TO SEE TE	PTCN0940
č	ALD AND NEW TANGENTS DIFFER IN SIGN OF LIN-TH COMPONENT.	PTCN0941
ŏ	IF SO, ATTEMPT TO COMPUTE A POINT XR BETWEEN XC AND XF	PTCN0942
Ë	FOR WHICH TANGENT COMPONENT VANISHES	PTCN0943
Ĉ		PTCN0944
C 1	<i>*******</i> *****************************	****PTCN0945
С		PTCN0946
c	IF (LIN.LE.O.UR.RSIEF.LE.O) BU IU 180	PICN044/
с С	CHECK EOD LIMIT INTERNAL	FILMV790
C C		DTCNADAO
<u> </u>	CHERT FOR LANAT ANTERVIE	PTCN0949 PTCN0950
С	TE (STGN(1.0.TC) TH).FO.STGN(1.0.TL) TH)) GO TO 140	PTCN0949 PTCN0950 PTCN0951
с с	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160	PTCN0949 PTCN0950 PTCN0951 PTCN0952
С С С	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS	PTCN0949 PTCN0950 PTCN0951 PTCN0952 PTCN0953
С С С С С С С	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS	PTCN0949 PTCN0950 PTCN0951 PTCN0952 PTCN0953 PTCN0954
С С С С С С	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GQ TQ 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM)	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0954 PTCN0955
с ссс с	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLN=ABS(TLLIM) IF (ATLLM.GT,0,5#ABSERR) GO TO 110	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0954 PTCN0955 PTCN0956
	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLN=ABS(TLLIM) IF (ATLLM.GT.0.5#ABSERR) GO TO 110	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0953 PTCN0955 PTCN0955 PTCN0957
0 000 000	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLN=ABS(TLLIM) IF (ATLLM.GT.0.5#ABSERR) GO TO 110 IF XC IS LINTT POINT, TL ALREADY CONTAINS TANGENT AT XC	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0955 PTCN0955 PTCN0955 PTCN0958 PTCN0958
0 000 000	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLN=ABS(TLLIM) IF (ATLLM.GT.0.STABSERR) GO TO 110 IF XC IS LINIT POINT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NUAP-PHOPK(IXC).1.PHOPK(IXP.1)	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0955 PTCN0955 PTCN0957 PTCN0958 PTCN0958 PTCN0959 PTCN0950
0 000 000	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT.0.S#ABSERR) GO TO 110 IF XC IS LINIF POINT, TL ALREADY CONTAINS TANGENT AF XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0955 PTCN0955 PTCN0957 PTCN0959 PTCN0950 PTCN0960 PTCN0960
000 000	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT.0,S#ABSERR) GO TO 110 IF (ATLLM.GT.0,S#ABSERR) GO TO 110 IF XC IS LIMIT PUINT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=ABS(TCLIM)	PTCN0949 PTCN0950 PTCN0952 PTCN0953 PTCN0954 PTCN0955 PTCN0955 PTCN0956 PTCN0958 PTCN0959 PTCN0959 PTCN0961 PTCN0961
0 000 000	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT,0,S#ABSERR) GO TO 110 IF XC IS LIMIT PUINT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=AB3(TCLIM) IF (ATCLIM=GT.0.5#ABSERR) GO TO 130	PTCN0949 PTCN0950 PTCN0952 PTCN0953 PTCN0954 PTCN0955 PTCN0956 PTCN0956 PTCN0958 PTCN0958 PTCN0959 PTCN0962 PTCN0962 PTCN0963
0 000 000	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT,0,5#ABSERR) GO TO 110 IF XC IS LINIF PU(NT, TL ALREADY CONTAINS TANGENF AF XC 100 CALL SCOPY(NVAR,RWORK((XC),1,RWORK((XR),1)) GO TO 310 110 ATCLIM=ABS(TCLIM) IF (ATCLIM.GT.0.5#ABSERR) GO TO 130 120 CALL SCOPY(NVAR,RWORK((XF),1,RWORK((XR),1))	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0954 PTCN0956 PTCN0957 PTCN0957 PTCN0959 PTCN0960 PTCN0961 PTCN0963 PTCN0963 PTCN0964
	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT,0,5#ABSERR) GO TO 110 IF XC IS LIMIT PUINT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=AB3(TCLIM) IF (ATCLIM.GT.0.5#ABSERR) GO TO 130 120 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1) CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1) CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1)	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0954 PTCN0956 PTCN0957 PTCN0959 PTCN0959 PTCN0960 PTCN0964 PTCN0963 PTCN0963 PTCN0965
	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT,0,STABSERR) GO TO 110 IF XC IS LINIF PUINT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=ABS(TCLIM) IF (ATCLIM.GT.0.STABSERR) GO TO 130 120 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1) CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1) CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXC),1) CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXC),1) GO TO 310	PTCN0949 PTCN0951 PTCN0951 PTCN0953 PTCN0953 PTCN0954 PTCN0956 PTCN0957 PTCN0957 PTCN0959 PTCN0960 PTCN0964 PTCN0965 PTCN0965 PTCN0965 PTCN0965
	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT.0,STABSERR) GO TO 110 IF XC IS LIMIT PUINT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK([XC),1,RWORK([XR),1) GO TO 310 110 ATCLIM=AB3(TCLIM) IF (ATCLIM.GT.0.STABSERR) GO TO 130 120 CALL SCOPY(NVAR,RWORK([XF),1,RWORK([XR),1) CALL SCOPY(NVAR,RWORK(IXF),1,RWORK([XR),1) CALL SCOPY(NVAR,RWORK(IXF),1,RWORK([XR),1) CALL SCOPY(NVAR,RWORK(ITC),1,RWORK([TL),1) GO TO 310	PTCN0949 PTCN0951 PTCN0952 PTCN0953 PTCN0953 PTCN0954 PTCN0956 PTCN0957 PTCN0957 PTCN0959 PTCN0960 PTCN0964 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0966 PTCN0967
	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT.0,S#ABSERR) GO TO 110 IF XC IS LINIT PUINT, TL ALREADY CONTAINS TANGENT AT XC 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 TEST FOR SNALL INTERVAL	PTCN0949 PTCN0951 PTCN0951 PTCN0953 PTCN0953 PTCN0954 PTCN0956 PTCN0957 PTCN0957 PTCN0959 PTCN0960 PTCN0964 PTCN0964 PTCN0965 PTCN0965 PTCN0966 PTCN0966 PTCN0968
	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT.0,STABSERR) GO TO 110 IF (ATLLM.GT.0,STABSERR) GO TO 110 IF XC IS LINIF POINT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=AB3(TCLIM) IF (ATCLIM.GT.0.STABSERR) 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 TEST FOR SMALL INTERVAL 130 YCLIM=PHODE (YCHLIM)	PTCN0949 PTCN0950 PTCN0952 PTCN0953 PTCN0953 PTCN0955 PTCN0956 PTCN0956 PTCN0956 PTCN0950 PTCN0960 PTCN0960 PTCN0963 PTCN0965 PTCN0965 PTCN0965 PTCN0967 PTCN0968 PTCN0968 PTCN0968
C CCC CCC CCC	<pre>IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT.0.STABSERR) GO TO 110 IF XC IS (INTF POINT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=AB3(TCLIM) IF (ATCLIM.GT.0.STABSERR) 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 TEST FOR SMALL INTERVAL 130 XCLIM=RWORK, XC+LIM) </pre>	PTCN0949 PTCN0950 PTCN0952 PTCN0953 PTCN0953 PTCN0954 PTCN0956 PTCN0956 PTCN0956 PTCN0959 PTCN0961 PTCN0963 PTCN0964 PTCN0965 PTCN0965 PTCN0967 PTCN0967 PTCN0967 PTCN0970 PTCN0970
0 000 000 000	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT.0.STABSERR) GO TO 110 IF XC IS LINIF POINT, TL ALREADY CONTAINS TANGENT AF XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=AB3(TCLIM) IF (ATCLIM.GT.0.STABSERR) 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 TEST FOR SMALL INTERVAL 130 XCLIM=RWORK, XC+LIM) DEL=ABS(XELIM-XCLIM)	PTCN0949 PTCN0950 PTCN0952 PTCN0953 PTCN0954 PTCN0956 PTCN0956 PTCN0956 PTCN0958 PTCN0959 PTCN0961 PTCN0963 PTCN0964 PTCN0964 PTCN0965 PTCN0965 PTCN0967 PTCN096971 PTCN0971 PTCN097
C CCC CCC CCC	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT,0,5#ABSERR) GO TO 110 IF XC IS LINIF PU(NT, TL ALREADY CONTAINS TANGENF AF XC 100 CALL SCOPY(NVAR,RWORK((XC)+1,RWORK((XR)+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(IXF)+1,RWORK(IXR)+1) CALL SCOPY(NVAR,RWORK(ITC)+1,RWORK(IXR)+1) GO TO 310 TEST FOR SMALL INTERVAL 130 XCLIM=RWORK, XC+LIM) XFLIM=RWORK(JXF+LIM) DEL=ABS(XFLIM-XCLIM)+ABS(XFLIM))	PTCN0949 PTCN0950 PTCN0952 PTCN0953 PTCN0953 PTCN0956 PTCN0956 PTCN0957 PTCN0957 PTCN0950 PTCN0960 PTCN0963 PTCN0964 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0967 PTCN0970 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977
C CCC CCC CCC	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLN=ABS(TLLIM) IF (ATLLX.GT.0.STABSERR) GO TO 110 IF XC IS LIMIT POINT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=ABS(TCLIM) IF (ATCLIM.GT.0.STABSERR) GO TO 130 120 CALL SCOPY(NVAR,RWORK(IXF).1,RWORK(IXR),1) CALL SCOPY(NVAR,RWORK(ITC).1,RWORK(IXR),1) GO TO 310 TEST FOR SMALL INTERVAL 130 XCLIM=RWORK(XCHIM) XABS=AMAX1(ABS(XCLIM),ARS(XFLIM)) IF (DEL.405((ABSERR+RELERRIXARS)) GO TO 140	PTCN0949 PTCN0950 PTCN0952 PTCN0953 PTCN0954 PTCN0956 PTCN0956 PTCN0957 PTCN0959 PTCN0960 PTCN0963 PTCN0964 PTCN0964 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0975 PTCN0970 PTCN0973 PTCN0973 PTCN0973 PTCN0973 PTCN0973 PTCN0973
C CCC CCC CCC	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT.0.5%ABSERR) GO TO 110 IF XC IS LINIF PUINT, TL ALREADY CONTAINS TANGENT AT XC 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(IXR),1) CALL SCOPY(NVAR,RWORK(ITC),1,RWORK(ITL),1) GO TO 310 TEST FUR SHALL INTERVAL 130 XCLIM=RWORK, XC4LIM) XABS=AMAX1(ABS(XCLIM),ARS(XFLIM)) IF (DEL,GT,(ABSERR+RELERR%ARS)) GO TO 140 IF (ATLLM.GT.ATCLIM) GO TO 120	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0956 PTCN0956 PTCN0957 PTCN0959 PTCN0960 PTCN0964 PTCN0964 PTCN0964 PTCN0964 PTCN0965 PTCN0967 PTCN0967 PTCN0973 PTCN0973 PTCN0975
C CCC CCC CCC	IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT,0,S#ABSERR) GO TO 110 IF XC IS LINIF PU(NT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=ARS(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 TU 310 TEST FUR SMALL INTERVAL 130 XCLIM=RWORK, XC+LIM) XFLIM=RWORK(JXF+LIM) DEL=ABS(XFLIM-XCLIM),ARS(XFLIM)) IF (DEL.GT.(ABSERR+RELERR#XARS)) GO FU 140 IF (ATLLM.GT.ATCLIM) GO TO 120 GO TU 100	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0956 PTCN0956 PTCN0957 PTCN0959 PTCN0959 PTCN0960 PTCN0963 PTCN0963 PTCN0968 PTCN0968 PTCN0968 PTCN0968 PTCN0968 PTCN0975 PTCN0973 PTCN0973 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975
C CCC CCC C	<pre>Intern for thirt interval IF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GQ TQ 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT.0,STABSERR) GO TQ 110 IF (ATLLM.GT.0,STABSERR) GO TQ 110 IF xc IS LIMIF PQ(NT, TL ALREADY CONTAINS TANGENF AF xc 100 CALL SCOPY(NVAR,RWORK((Ixc),1,RWORK((IXR),1)) GD TQ 310 110 ATCLIM=AB3(TCLIM) IF (ATCLIM.GT.0.STABSERR) GO TO 130 120 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1) CALL SCOPY(NVAR,RWORK(ITC),1,RWORK(IXR),1) GO TQ 310 TEST FOR SMALL INTERVAL 130 xCLIM=RWORK.XCH(M) xfLIM=RWORK(XFFLIM) DEL=ABS(XFLIM=XCLIM) xABS=AMAX1(ABS(XCLIM)+ABS(XFLIM)) IF (DEL.GT.(ABSERR+RELERRTXARS)) GO FO 140 IF (ATLLM.GT.ATCLIM) GO TO 120 GO TO 100</pre>	PTCN0949 PTCN0950 PTCN0952 PTCN0953 PTCN0953 PTCN0955 PTCN0956 PTCN0957 PTCN0957 PTCN0959 PTCN0961 PTCN0963 PTCN0965 PTCN0965 PTCN0965 PTCN0966 PTCN0967 PTCN0975 PTCN0973 PTCN0973 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975 PTCN0975
0 000 000 000 000 000	<pre>India for Linit Interval IF (SIGN(1.0,TCLIN).EG.SIGN(1.0,TLLIN)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIN) IF (ATLLM.GT,0,STABSERR) GO TO 110 IF XC IS LINIF PU(NT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=ABS(TCLIN) IF (ATCLIM.GT.0.STABSERR) GO TO 130 120 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1) CALL SCOPY(NVAR,RWORK(ITC),1,RWORK(IXR),1) CALL SCOPY(NVAR,RWORK(ITC),1,RWORK(ITL),1) GO TO 310 TEST FOR SMALL INTERVAL 130 XCL(M=RWORK, XC+LIN) XABS=AMAX1(ABS(XCLIN),ARS(XFLIN)) IF (DEL,GT.(ABSERR+RELEXRIXARS)) GO TO 140 IF (ATLLM.GT.ATCLIN) GO TO 120 GO TO 100 BEGIN ROOT-FINGING ITERATION WITH INTERVAL (0,1) AND ENDELSENTERVAL (0,1) AND END</pre>	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0955 PTCN0956 PTCN0956 PTCN0956 PTCN0959 PTCN0950 PTCN0961 PTCN0963 PTCN0964 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0967 PTCN0973 PTCN0973 PTCN0974 PTCN0976 PTCN0976 PTCN0977 PTCN0977 PTCN0977 PTCN0977
0 000 000 000 000 000	<pre>India for Linit Interval IF (SIGN(1.0,TCLIN).EQ.SIGN(1.0,TLLIN)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATULN=ABS(TLLIN) IF (ATULM.GT,0,STABSERR) GO TO 110 IF XC IS LINIF PU(NT, TL ALREADY CONTAINS TANGENF AF XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=AB3(TCLIN) IF (ATCLIM.GT.0.STABSERR) 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 TEST FOR SMALL INTERVAL 130 XCL(M=RWORK, XC+L(N) XFLIM=RWORK, XC+L(N) XABS=AMAX1(ABS(XCLIN).ABS(XFLIN)) IF (DEL,GT.(ABSERR+RELERRIXARS)) GO TO 140 IF (ATLLM.GT.ATCLIN) GO TO 120 GO TO 100 BEGIN ROOT-FINOING ITERATION WITH INTERVAL (0,1) AND FUNCTION VALUES TL(LIM), TC(LIM).</pre>	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0954 PTCN0955 PTCN0955 PTCN0956 PTCN0957 PTCN0959 PTCN0959 PTCN0961 PTCN0963 PTCN0963 PTCN0964 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0967 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977
0 000 000 000 000 0000	TF (SIGN(1.0,TCLIM).EQ.SIGN(1.0,TLLIM)) GO TO 160 TEST FOR ACCEPTABLE ENDPOINTS ATLLM=ABS(TLLIM) IF (ATLLM.GT.0,STABSERR) GO TO 110 TF XC IS LIMIT PU(NT, TL ALREADY CONTAINS TANGENT AT XC 100 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) GO TO 310 110 ATCLIM=ABS(TCLIM) IF (ATCLIM.GT.0.STABSERR) GO TO 130 120 CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) CALL SCOPY(NVAR,RWORK(IXC),1,RWORK(IXR),1) CALL SCOPY(NVAR,RWORK(ITC),1,RWORK(IXR),1) GO TO 310 TEST FOR SMALL INTERVAL 130 XCLIM=RWORK, XC4LIM) XFLIM=RWORK, XC4LIM) AGBS=AMAX1(ABS(XCLIM),ARS(XFLIM)) IF (DEL.GT.(ABSERR+RELERRXXARS)) GO TO 140 IF (ATLLM.GT.ATCLIM) GO TO 120 GO TO 100 BEGIN ROOT-FINOING ITERATION WITH INTERVAL (0.1) AND FUNCTION VALUES TL(LIM), TC(LIM).	PTCN0949 PTCN0950 PTCN0951 PTCN0953 PTCN0953 PTCN0956 PTCN0956 PTCN0956 PTCN0958 PTCN0958 PTCN0958 PTCN0958 PTCN0961 PTCN0964 PTCN0964 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0965 PTCN0970 PTCN0973 PTCN0975 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977 PTCN0977

140 KOUNT=0 PTCN0981 PTCHOV82 A=0.0 PTCN0983 FA=TLL IN PTCN0984 TSN=TCLIN PTCN0985 C=1.0 FC=TCI.IM PTCN0986 C PTCN0987 С SET VPLIN TO THE INDEX OF HAXINUM ENTRY OF SECANT PTCN0988 (EXCEPT THAT IPLIM MUST NOT EQUAL LIM) AND SAVE THE SIGN OF THE MAXIMUM COMPONENT IN DIRLIM SO THAT NEW TANGENTS MAY BE PROPERLY SIGNED. Ĉ PTCN0989 PTCN0990 č PTCN0991 PTCN0992 CALL SCOPY(NVAR; RWORK(IXF); 1; RWORK(JXR); 1) PTCN0993 CALL SAXPY(HUAR,-1,0,RHORK(TXC),1,RHORK(TXR),1) PTCN0994 RWORK(JXR+LIM)=0.0 PTCN0995 IPL IN= ISAMAX (HUAR + RHORK (TXR) + 1) PTCNO99A DIRLIM=SIGN(1.0,RWORK(JXR+IPLIM)) PTCN0997 PTCN0998 CALL ROOTFINDER FOR APPROXIMATE ROOT SN, SET X=SN*XF+(1-SN)*XC Call Corrector to return to curve on line X(tpl[m)=constant, compute tangent there, and set function value to tangent(lim) PTCN0999 0000 PTCN1000 PTCN1001 PTCN1002 150 CALL ROOT(A+FA+SN+TSN+C+FC+KOUNT+IFLAG) PTCN1003 NI_MRT=NL MRT+1 PTCN1004 IF(IFLAG.LF.-1)60 T0 350 IF(IFLAG.EP.-1.0R.IFLAB.EP.0)60 T0 310 CALL SCOPY(NVAR,RWORK(IXF),1,RWORK(IXR),1) PTCN1005 PTCN1006 PTCN1007 CALL SSCAL (NVAR, SN, RWORK(1XR), 1) PTCN1008 SCALER=1.0-SN CALL SAXPY(NVAR+SCALER+RWORK(IXC)+1+RWORK(IXR)+1) CALL CORECT(NVAR+RWORK(IXR)+TPLIN+RWORK(ITL)+TERR+THOO+RWORK(TFP) PTCN1009 PTCN1010 PTCN1011 1 . TPUT . ABSERR . RELERR . XSTPLM . NEQN . FNRM) PTCN1012 NLHCR-ALACR+KN IF (IERR.NE.0) GD TO 350 PTCN1013 PTCN1014 TCALL=1 PTCN1015 IPT=IPL IN PTCN1016 CALL TANGN F (NUAR, RWORK (IXR), IPT, RWORK (ITL), IRET, ICALL, 1 RWORK (IFP), IPVT, NEGN, DETLIN, IEXLIN) PTCN1017 PTCN1018 IF(IRET.LT.0)60 TO 350 PTCN1019 PTCN1020 č ADJUST THE SIGN OF THE TANGENT SO THAT THE IPLIN-IH COMPONENT HAS THE SAME SIGN AS THE IPLIN-TH COMPONENT OF THE SECANT PTCN1021 PTCN1022 č PTCN1023 JF (BIRLIM.NE.SIGN(1.0,RWORK(JTL+JPLIM))) PTCN1024 1 CALL SSCAL (NYAR, -1,0, RWORK(TTL),1) PTCN1025 PTCN1026 SEE IF WE CAN ACCEPT THE NEW POINT BECAUSE TANGENT(LIM) IS SMALL OR MUST 'ACCEPT' THE POINT RECAUSE THE VALUES ARE NOT DECREASING RAPIDLY ENOUGH, OR IF WE CAN GO ON. Ĉ PTCN1027 000 PTCN1028 PTCN1029 PTCN1030 PTCN1031 TSHOL D=TSN TSN=RWORK(JTL+LIM) PTCN1032 1F (A8S(TSN), LE. 0. 5#A8SERR) GO TO 310 PTCN1033 60 10 150 PTCN1034 PTCN1035 **PTCN1036 PTCN1037 č PTCN1038 5. STEP LENGTH COMPUTATION. COMPUTE STEPLENGTH HTANCF PTCN1039 CCCCCCC PTCN1040 THE FORMULAS UNDERLYING THE ALGORITHM ARE PTCN1041 PTCN1042 LET PTCN1043 ALPHLC = MAXIMUM OF ARCCOS(TL;TC) AND ALFMIN = 2#ARCCOS(1-EPMACH) PTCN1044 č HSECI.C = HORM (XL-XC) PTCN1045 Č HSECLI. + NORM (XL-XLL) PTCN1046 = ARS(STN(,5#ALPHLC)) PTCN1047 ARSTN CURVLC = LAST VALUE OF CURVCF CURVCF = 2*ABSIN/HSECLC CORDIS = OPTIMIZED CORRECTOR DISTANCE TO CURRENT CONTINUATION POINT. PTCN1050 C

C UNLESS (CORDIS, EQ. 0, 0); BECAUSE THE PREDICTED POINT WAS PTO C UNLESS (CORDIS, EQ. 0, 0); BECAUSE THE PREDICTED POINT WAS PTO C INMEDIATELY ACCEPTED. IN SUCH A CASE, SET HTANCF=HSECLC PTO C INSTEAD OF USING FIRST ESTIMATE FOR HTANCF. PTO	N1051 N1052
C INMEDIATELY ACCEPTED. IN SUCH A CASE, SET HTANCF=HSECLC PTC C INSTEAD OF USING FIRST ESTIMATE FOR HTANCF. PTC	N1053
C INSTEAD OF USING FIRST ESTIMATE FOR HTANCE. PTO	MIGEA
	CN1055
	201057
Č CURVXF = CURVCF + HSECLC*(CURVCF-CURVLC)/(HSECLC+HSECLL) PTC	N1058
C BUT CURVEF HUST BE GREATER THAN . 001, AND A SIMPLER FORMULA IS USED PTO	CN1059
C IF WE DO NOT HAVE DATA AT TWO PREVIOUS POINTS. PTO	CN1060
C FIRST ESTIMATE: (UNLESS (CORDIS.E0.0.0))	CN1062
Č PTČ	N1063
C HEAHOF = SURT(2*CORDIS/CURYXE) PTO	CN1064
	501060 X1044
C PTC	N1067
C HTANCF = HTANCF*(1.0 + HTANCF*(TC(JPC)-TL(JPC))/(2*HSECLC*TC(JPC))) PTC	N1068
C PEAR HICTMENT AND TRIMCATIONS!	CN1069
	N1071
C 1F STEPSIZE REDUCTION OCCURRED DURING LAST CORRECTOR PROCESS, PTC	N1072
C HTANCE IS FORCED TO BE LESS THAN (HEACT-1)*HSECLC/2. PTG	N1073
	N1074
C HIANCE HUSI LIZ BEINEZN (HSECCC/HEACI) AND (HSECCCCHEACI). PIC	581078 1074
C HIANCE IS ALWAYS FORCED TO LIE BETWEEN HILN AND HHAX.	N1077
C PTC	N1078
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	N1079
U 1910 C 1920 - 1920 - 1920 - 1920 - 1920 - 1920 - 1920 - 1920 D 1920 - 1920 - 1920 - 1920 - 1920 - 1920 - 1920 - 1920	N1080
C UN FIRST STEP, USE HTANEFEN. PTC	N1082
C TF PREVIOUS STEP WAS OF SIZE ZERO, USE STEPSTZE HDEF=(HNTN+HNAX)/2 PTC	N1083
	CN1084
160 (F (KSTEP,GT,O.ANU.HSELLC.GT,O,O) 50 10 1/0 PTC	CN1085
TF(KSTEP.LE.O)HTANCE=H PTC	N1087
60 TO 190 PTC	880 14
170 IF (ALPHLC) LT ALPHIN ALPHIC = ALFHIN PTO	N1089
AFSIN=ABS(SIN(.5TALPHLC))	N1090
C COMPLITE NEW CURVATURE DATA PTC	N1092
C PTC	N1093
CURVLC=CURVCF PTC	N1094
CURVCF=2,0xABSTN/HSECLC PTC	CN1095
	N1070
1 CURVXF=CURVCF+HSECLC*(CURVCF-CURVLC)/(HSECLC+HSECLL) PTC	CN1098
CURVXF=AMAX1(CURVXF,TENN3) PTC	N1099
IF (IWRITE, GE, 2) WRITE (&, 560) CURVCE, CURVXE PTC	CN1100
L C IF THE CONVERGENCE DISTANCE IS ZERO. SET FIRST ESTIMATE TO ASECIC. PTU	N1102
C OTHERWISE, TRUNCATE CORDIS TO LIE BETWEEN JOI #HSECLC AND HSECLC. PTO	N1103
C PTC	CN1104
HTANCF#HSECLC PTC	N1105
TENP#TENNO#NSECIC PTC	N1107
CORDIS=AMAX1(CORDIS,TEMP) PTC	N1108
CURDIS=ANTH1 (CORDIS+HSECLC) PTC	N1109
	N1110
C PARAMETER RIRECTION, THEN TRUNCATE POR CONVERTE PROVIDENCE OF CONTINUES PROVIDENCE OF CONTRACTION PROVIDENCE OF CONTRACT PROVIDACT PROVIDACT PROVIDACT PROVIDE	2N1112
C	N1113
HTANCF=SORT(2.0*CORDIS/CURVXF) PTC	N1114
180 IF (NRED,GT,O)HTANCF*ANIN1 (HTANCF+(NFACT-1,O)#HSECLC#,5) PTC	N1115
UNUNDELIVUUT (IVUUT VILITU/VILITU/VILE(VDANNUP)/VINLE(MSECLU) PIL HTANYF HTANPS ZSARI (NAN UIS) OT	N1117
TENP=HSECLC#HRED PTC	N1118
HTANCE = ANAX1 (HTANCE, TENP) PTC	N1119
TENP=HSECLC#HFACT PTC	N1120

AND STREET

ſ		нт Нт Нт	an An An	CF: CF:	= A† = A† = A†	ITN IAX ITN	1(1(1(HT HT HT	ani Ani Ani	CF CF	, TE , HM , HM	: MF (() (A)	2) () ()						-	•											PT PT PT	CN CN CN	11 11 11	21 22 23
0000000000	(*** 6. AS JTI PU FAI	222 81 87 87 87 87 87 87 87 87 87 87 87 87 87	RE AR TI	2 2 1 0 1 (7 T) 0 N 0 N	8 # 4 5 T 1 4 G 1 T 1 T 2 W	IN PO IF RY HM	**	XX NR CR GA	**: Ci EC' TN	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	XXX REC Rec Faj	144 111 112 112 112	10N XR 37 Rec	** * * * *	(#) (1) (0)	### EP9 Th UCE N \		144 50 516 1.	:**: ISJI ILI. IPS: INI	8 # 1 05 1 7 E 2 Y	XR XR E BE	1##: = X 198 198 198	##: [F] [BA] [BA]	*** hta te: for nd(NCI NCI PI	***) **T(==================================	2 # 1 C D N J C 1 F - 9	K## For Ste	*** PST	*** 7e	PT PT PT PT PT PT	CN CN CN CN CN CN CN		256789012
сл С	***	***	**	**	k X X	**	**	**	**	k k :	**1	**1	***	**	1	***	**	**	**	**1	***	**	**:	***	**1	***	**1	K**:	***	***	PT	CN:		33
U	190 200 210	KS NR CA TF	TE TE LL LL	P()= P=) =0 \$(\$(#R)	=KS (S1 COP AXP [TP	ТЕ ГЕР РУ (РУ (P +1 HUN E,	AR AR 2)	⇒RI ∍H° ⊌R	10) 14 17	RK (NCF	([) [} [}	(F) RWO 570	+ 1 RK	() (() T#	RWC I T C A N C)RX ;> ; ;F		XR RWI) + 1]R	() ()	XR),	1)							PT PT PT PT PT			333333340
C		IF CA 1 1 NC IF IF	(1). PV NC (WR CI T I R TI R TI I EI I EI	ITE DRE ABS ABS ABS AC) RR RR RR	E.G ECT ECT ECT ECT ECT ECT ECT ECT ECT ECT	E(R+0-	3) YAE })	4R R = 1 L = 1 G 0 G 1	TTI Rini Rir T	E (& DRK 9 X S 0 2 TO	6,5 (1) 6TE 23(580 [XR EP 20))(;)• NE	(R I T J C I	NOR PCJ NJF	RK (Rk NF	(JX 108 (M)	(R+) (F)	[) ([T]	•] = .) •	=1 + (TE)	NV(RR	AR) ≠ [∦	00	RW	ORI	< T	FP)	,	PT PT PT PT PT PT		51 11 11 11 11	41 42 43 44 45 46 45
č	0K	ĊŰ	NV	ER)E)	ICE	,	TR	Y	A	SMA	M.1	.88	9	ST	EPS	517	Ε.													PT	CN I	ii	48 49
v	220	HT IF NR IF GU CA	AN (ED () Ti () LL	CF: HT(= N) IE(S(S) S(S)	HY ANC RR 200 AXF SC/	120 120 120 120 120 120 120 120 120 120		17A • H • 2) • AR	NC: NT: G(,-:	F N) D 1 (QC TO 0,F 0,F	22	TO 20 DRK DRK	39	20 [X]	C), R),	1,	R	IQRI	()		<>>+	1)								PT PT PT PT PT PT			701234567
С		Ğ	Ť	0	210)` `			, ,	••	7		יחי			/ • 1			, NN		1		,								PT PT	CN:	11 11	58
ក្ត័រ	***	***	**	**:	**1	***	**	**	**:	FX :	***	121	***	***	t tr	***	:*1	***	**	K # 1	***	**	**:	***	**1	t##1	***	***	***	***	ΡŤ	CN	ļi	ÃÔ
000000000000000000000000000000000000000	7, UP) C00 A C00 (4 SE	S DAT NPU FAC RRS FC E S		CE: OLI CI R ONI FUI ERI	SSP DA DR DR DR DA THE D THE D THE D THE D THE D		S C N A D E	TE UR TH DE DN	P RE CH STI HE	RA IO	STQ CC ZE ESC BLS FC ER		E T TIN ES NUM HO			RNA IGN COR DI OF IEI THY		(0) (01 (01 (0) (0) (0) (0) (0) (0) (0) (0) (0) (0)	81 108 108 108 108 100 100	EF(5. 51 04 01 00	DRE IEP NLU DR	E RI IÈ I ISTI CT	ETI Ci WH EP: T,	URN DMP ICH S		DULI	0				PTPTPT			06666666666666666666666666666666666666
C		447 MC		44) (114.		144 196	-#-# 		44. DC	₩.₩.	**1	. # 4	K # #			* * 4			. 4. 4				* * ·	***	. 4. 3. 1	L H. K .)	F 4 1		***	***	PT			71
0000	CO SE	IF MPL T (TE DR	07: RE) C(DT)	D.) DRF 5 -	NE REC	0. TO AX	AN R	D. STI	ÏV EP 4	R I 1 = 0F	ΓΕ - χ(¢(.GE C+H DRR	:.2 Ita 1ec	?) \\\(\; T	WRI CF4 UR	LTE Et(57	E (8 ;-) TE P	(F	90)	NR	RED									PT PT PT PT	CN CN CN		/7777777777777777777777777777777777777
-		CA CA CA IN CA IN	11. 11. 14. 14. 14. 14. 14. 14. 14. 14.	50 50 51 51 51 51 51 15	CDF AXI SAI SAI EQ	PY(PY(14X 85(.0)	NY NY NY RI RI	iar Var Var Var Var Var Var Var Var	• R • - H • H • K • C I	40 1 - 1 7 A 1 7 A 1 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7	RK (0 + F NC F 0 R M C + 1 0 - ((T) RW(8 + H (() []])	(F) DRM RW() IXC AX)	; 1 ((1 RK :);	/ [X1 ((1	R10 R) ((T))) R K 1 1 1	(() R) 1	XC DRI RW) + 1 K (] ORM	() () ()	() + (XC	1)),	1)							PT PT PT PT PT PT	CN CN CN CN CN		7878818888
C C C	MO De	DIF SIR	Y	100 14	RD I UMI	is Ber	тØ	I A IF	v coi	AL RR	UE Ect	T) FOI		L TE	101 191	UI. I S) (:0R	RE	SPC	םאו	T	0	THE							PT PT	UN) CN) CN		84 85 86
Ĉ			(C IEG IET	0R A= A=(019 XS1 0,0	B » E I E P)	0, /(0, OR	0)(DJ	50 5	τı)	300	•																	PT PT PT	CN CN CN		87 88 89 90

PTCN1191 PTCN1192 IF(INOD, EQ.1) 60T0 260 Č FULL NEWTON METHOD FOR CORRECTOR STEPS PTCN1193 PTCN1194 IF (KN.LE.1) THETA=8.0 IF (KN.ER.4) THE FA=1.0 PTCN1195 PTCN1196 IF (THETA.NE.0.0)60 TO 290 PTCN1197 IF(KN.GT.4)G0 T0 240 PTCN1198 PTCN1199 LK=4*KN-7 THETA=1.0 PTCN1200 IF (OMEGA.GE, WRGE(LK))GOTO 290 PTCN1201 PTCN1202 151-11 IF (OHEGA.GE.WRGE(LK+1))GOTO 250 PTCN1203 PTCN1204 IST=LK+2 PTCN1 205 IF (OMEGA.GE.WRGE(LK+2))GOTO 250 PTCN1206 PTCN1207 THETA=8.0 GOTO 290 240 THETA=0.125 PTCN1208 IF (KN.6E.7) GOTO 290 PTCN1209 PTCN1210 I.K=4#KH-16 PTCN1211 PTCN1212 IF(OMEGA.LE.WRGE(LK))GOTO 290 LST=2±KN-1 250 THETA=ACOF(LST)+ACOF(LST+1)#ALOG(OMEGA) PTCN1213 PTCH1214 GOTU 290 PTCN1215 Ĉ MODIFIED NEWTON METHOD FOR CORRECTOR STEPS PTCN1216 Ĉ PTCN1217 260 TF (KH.LE.1) THETA=8.0 PTCN1218 PTCN1219 PTCN1270 IF (KN. EQ. 10) THETA=1.0 TF(THETA, NE.0,0)60 TO 290 EXPON=FLOAT(KN-1)/FLOAT(KN-10) PTCN1221 PTCN1222 PTCN1223 PTCN1224 AVOID OVERFLOW OR UNDERFLOW BY ANTICIPATING CUTOFF VALUES OF THETA Ĉ C PTCN1225 PTCN1226 TF (KN.GT.10) 60 TO 270 (8.0**EXPON.GT.OHEGA) THETA=8.0 (.125**EXPON.LT.OHEGA) THETA=.125 PTCN1227 TF PTCN1228 ĪF PTCN1229 PTCN1230 IF (THETA, NE. 0.0) 60 TO 290 GO TO 280 270 IF (8.0**EXPON.LT. OMEGA) THETA=8.0 IF (,125**EXPON.GT.OMEGA) [HETA=,125 PTCN1231 PTCN1232 IF (THETA, NE. 0.0) GD TO 290 PTCN1233 280 EXPON=1.0/EXPON THETA=QHEGA**EXPON PTCN1234 PTCN1235 THETA=AMAX1(THETA,0.125) PTCN1236 PTCN1237 THETA=AMIN1(THETA,8.0) PTCN1238 SET THE MODIFIED VALUE OF CORDIS PTCN1239 PTCN1240 PTCN1241 PTCN1242 290 CORDIS=THETA*CORDIS IF([WRTTE:GE,2)WRTTE(6,600)DMEGA,THETA;CORDIS 300 CALL SCOPY(NVAR;RWORK(IXF),1;RWORK(IXC),1) CALL SCOPY(NVAR;RWORK(IXR),1;RWORK(IXF),1) PTCN1243 PTCN1244 GD TO 340 PTCN1245 C PTCN1244 PTCN1248 C Ĉ SET VALUE OF IRET. IF AN ERROR OCCURRED, PRINT RETURNS. PTCN1249 £ PTCN1253 PTCN1254 C С RETURN LINIT POINT PTCN1255 PTCN1256 C PTCN1257 PTCN1258 310 JRET#2 RETURN £ PTCN1259 RETURN WITH TARGET POINT PTCN1260 C

51

21.44.29

PTCN1261 PTCN1262 С 320 (RET=1 PTCN1263 RETURN PTCN1264 PTCN1265 С RETURN WITH CONTINUATION POINT PTCH1266 330 CALL SCOPY(NVAR; RWORK(IXF); 1; RWORK(IXR); 1) PTCN1267 340 TRET=0 PTCN1268 PTCN1269 H=HTANCF PTCN1270 RETURN PTCN1271 PTCN1272 C C ERROR RETURNS C PTCN1273 350 IRET=-1 PTCN1274 IF (IWRITE, GE, 1) WRITE (6,450) RETURN PTCN1275 PTCH1274 360 IRET=-2 PTCN1277 TF(IWRTTE,GE,1)WRITE(6,460) PTCN1278 RETURN PTCN1279 370 IRET=-3 PTCN1280 PTCN1281 PTCN1282 IF(IWRITE.GE.J)WRITE(6,470) RETURN 380 IRET=-4 PTCN1283 TF([WRITE,GE,1)WRITE(6,480) **PTCN1284** RETURN PTCN1285 390 IRET=-5 PTCN1286 IF (INRITE, GE, 1) WRITE (6, 490) HTANCF, HHIN PTCN1287 RETURN PTCN1288 400 JRET=-0 PTCN1289 TF(IWRITE,GE,1)WRITE(6,500) PTCN1290 RETURN PTCN1291 410 IRET=-7 PTCN1292 IF(IWRITE.GE.1)WRITE(6,510) PTCN1293 RETURN PTCN1294 PTCN1295 PTCN1296 420 JRET=-8 TF(TWRITE,GE,1)WRITE(6,520) RETURN PTCN1297 430 TRET=-9 PTCN1298 IF(IWRITE,GE(1)WRITE(6,530) PTCN1299 RETURN PTCN1300 440 IRET=-10 PTCN1301 IF(TWRITE, GE, 1) WRITE(6,540) NUAR, ISIZE PTCN1302 RETURN PTCN1303 450 FORMAT(26HOLINIT POINT FINDER FAILED) PTCN1304 460 FORMAT(SOHOCORECT, SEEKING TARGET POINT, TOOK TOO MANY STEPS) 470 FORMAT(SOHOCORECT, SEEKING TARGET, CALLED SOLVE WHICH FALLED) PTCN1305 PTCN1306 480 FORMAT(45HOCORECT, SEEKING TARGET, FAILED WITH RAD STEP) PTCN1307 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) PTCN1308 PTCN1309 PTCN1310 520 FORMAT(33HOSOLVE FAILED IN CALL FROM CORECT) 530 FORMAT(23HOTANGENT VECTOR IS ZERO) 540 FORMAT(26HOUNACCEPTABLE JNPUT NVAR=,110,7H ISJZE=,110) 550 FORMAT(36H ANGLE RETWEEN 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 ,12,16H STEP REDUCTIONS) 600 FORMAT(7H OMEGA=,F12.5,7H THETA=,F12.5,9H NEW RAD=,F12.5) 610 FORMAT(31H TANGHT ANTICIPATES L(NIT POINT) PTCN1311 PTCN1312 PTCN1313 PTCN1314 PTCN1315 PTCN1316 PTCN1317 PTCN1318 PTCN1319 PTCN1320 PTCN1321 *PTCN1322 PTCN1323 С PTCN1324 END

	SUBROUTINE CORECT(NVAR,X,JHOLD,WORK,IERR,IHOD,FPRYH,IPUT,	CRETOOOT
~	1 ARSERR + RELERR + XSTEP + NEQN + FNRM)	CRCT0002
Сж	******	CRU10003
č	1	CRCT0005
Č	SUBROUTINE CORECT PERFORMS THE CORRECTOR ITERATIONS ON A STARTING	CRCT0006
C C	POINT. THE CURRECTION METHOD IS EITHER FULL (IMOD=0) OR MODIFIED (IMOD=1) NEWTON'S METHOD FOR MODIFIED NEWTON'S	CRCT0007
č	HETHOR, THE JACOBIAN IS TO BE EVALUATED ONLY AT THE STARTING POINT.	CRCT0008
č	IF B IS THE VALUE OF X(IHOLD) FOR THE INPUT STARTING POINT,	CRCT0010
C	THEN THE AUGMENTING EQUATION IS X(IHOI,D)=B; THAT TS, THE THOID-TH COMPONENT OF Y IS TO BE HELD ETYCD	CRCT0011
č	THE AUGHENTED SYSTEM TO BE SOLVED IS THEN DEA(X, IHOLD) #DELTA=EA(X)	CRCT0013
Č		CRCT0014
C	INPUT Y - THE STARTING ROINT COD THE CONDECTOR ITERATION	CRCT0015
ň	THOLE = COMPONENT OF X THAT WILL NOT BE CHANGED DURING ITERATION	CRCT0017
Ĉ	IMOD = FLAG FOR TYPE OF NEWTON'S METHOD TO BE USED.	CRCT0019
Č	WHEN INON=0, JACOBIAN IS TO BE EVALUATED AT EVERY	CRCT0019
č	IF INOD=1, THE JACOBIAN IS ONLY EVALUATED AT THE STARTING	CRCT0021
Ĉ	POINT, AND KNHAX IS SET TO 20.	CRCT0022
C	OUTDUT	CRCT0023
č	X = SOLUTION VECTOR ON A SUCCESSFUL CALL TO CORECT.	CRCT0025
ē	WORK = THE RESIDUAL F(X), AFTER A SUCCESSFUL CALL TO CORECT.	CRCT0026
Č	IERR = THE RETURN FLAG WITH THE FOLLOWING VALUES	CRCT0027
C C	-2 HRALHUH MUHRER OF CORRECTOR THERBILONS WERE THREAD	CRCT0028
č	O SUCCESSFUL CORRECTION. VECTOR X RETURNED SATISIFES	CRCT0030
Ç	ABS(F(X)).LE.ARSERR	CRCT0031
C C	KN = THE NUMBER OF CORRECTOR ITERATIONS TAKEN ON THIS CALL	CRCT0033
č		CRCT0034
Ç	THIS SUBROUTINE IS CALLED BY	CRCT0035
C C	AND CALLS	CRCT0037
č	SOLVE	CRCT0038
Č	FORTRAN ABS	CRCT0039
C C	LINPAR ISANAX	CRCT0040
č	USER FCTN	CRCT0042
C.		CRCT0043
-U≢ C	***************************************	CRCT0045
-	REAL X(NVAR), HORK(NVAR), FPSYH(NVAR, NVAR)	CRCT0046
	INTEGER JPVT (NVAR)	CRCT0047
	COMMON /COUNTY ILESLITINGLINGICEINGICEINERCEINERCHINGEINER	CRCT0049
	COMMON /OUTPUT/ IWRITE	CRCT0050
-	CONNON /TOL/ EPHACH;EPSATE;EPSQRT	CRCT0051
C	INITIALIZE	CRCT0052
č		CRCT0054
		CRCT0055
	KRMAX=10 TF(THOD_FQ.1)KNHAX=20	CRCT0057
	IERR=0	CRCT0058
	FMP=2.0	CRCT0059
	XSTEP=0.0	CRCT0061
	CALL FCTN (NVAR, X, WORK)	CRCT0062
	IFEVAL=IFEVAL+1	CRCT0063
	LINA=LONNALINE (NETWORKIT) FNRH=ARS(UORK(IMAX))	CRCT0065
	WORK (NVAR)=0.0	CRCT0066
č	ATOMATED ABSEND TEST ON CTADITING POPUL	CRCT0067
č	SIKTUTER MAGERK TEST AN STAKITUR LATUL	CRCT0069
-	IF (FNRM.LE.0.5#ARSERR) GO TO 60	CRCT0070

CRCT0071 Ĉ ITERATION LOOP CRCT0072 CRCT0073 DO 20 I=1, KNMAX **CRCT0074** KN=I CRC10075 CALL SOLVE(NVAR, X, WORK, IHOLD, DETA, IEXP, IERR, ICALL, IHOD, FPRYM, CRCT0076 IPVT) CRCT0077 1 ICRSL=ICRSL+1 **CRCT0078** IF(JMOD.EQ.1)JCALL=0 **CRCT0079** ISOLVE=ISOLVE+1 CRCT0080 IF (IERR.NE.0) GO TO 50 CRCT0081 FNRML =FNRM **CRCT0082** XSTEPL=XSTEP CRCT0083 CALL SAXPY(NVAR,-1.0,WORK,1,X,1) CRCT0084 IMAX=ISAMAX(NVAR,WORK,1) XSTEP=ABS(WORK(IMAX)) CRCT0085 CRCT0086 IMAX=ISAMAX(NVAR,X,1) XNRM=ABS(X(IMAX)) CRCT0087 CRCT0088 CALL FCTN(NVAR,X,WORK) CRCT0089 IFEVAL=IFEVAL+1 CRCT0090 IMAX=ISAMAX (NEQN, WORK, 1) CRCT0091 FNRH=ARS(WORK([HAX)) WORK(NVAR)=0.0 CRCT0092 CRCT0093 С CRCT0094 Ć ACCEPTANCE TEST CRCT0095 C CRCT0096 IF (FNRH. I E. EPSATE) GO TO 60 CRCT0097 IF (FNRM.GT.ABSERR) GO TO 10 CRCT0098 (XSTEP.LE. (ABSERR+RELERR*XNRM)) GO TO 60 **TF** CRCT0099 C **CRCT0100** REJECTION TEST CRCT0101 č **CRCT0102** IF (KN.GT.1.AND.XSTEP.GT.(FMP*XSTEPL)) GO TO 30 IF (FNRM.GT.(FMP*FNRHL)) GO TO 30 10 CRCT0103 CRCT0104 FMP=1.05 20 **CRCT0105** GO TO 40 CRCT0106 C CRCT0107 UNSUCCESSFUL STEP C CRCT0108 ČRČŤŎĨŎŸ 30 IERR=-3 CRCT0110 IF(IWRITE.E0.2)WRITE(6,120) CRCT0111 GO TO 70 CRCT0112 CRCT0113 C F.F.T0114 MAXIMUM NUMBER OF CORRECTOR STEPS REACHED LRCT0115 CRCT0116 CRCT0117 40 IERR=-2 IF(IWRITE.ER.2)WRITE(6:110) GO TO 70 CRCT0118 C CRCT0119 ERROR RETURN IN SOLVE C **CRCT0120** С CRCT0121 50 IERR=-1 **CRCT0122** IF(IWRJTE, ER. 2)WRITE(6,100) **CRCT0123** 60 TO 70 **CRCT0124** C CRCT0125 Ċ SUCCESSFUL STEP CRCT0126 £ CRCT0127 60 IERR=0 CRCT0128 70 KNSUM=KNSUH+KN CRCT0129 IF(IWRITE.EQ.2)WRITE(6,80) KN,XSTEP IF(IWRITE.EQ.2)WRITE(6,90)IHOLD CRCT0130 **CRCT0131** CRCT0132 CRCT0133 RETURN 80 FORMAT(13H CORECT TOOK ,12,21H STEPS, LAST ONE WAS ,E12.5) 90 FORMAT(14H CORECT IHOLD=,13) **CRCT0134** 100 FORMAT(31HOSOLVE FAILED, CALLED BY CORECT) 110 FORMAT(25HOTOD MANY CORRECTOR STEPS) 120 FORMAT(24HOCORRECTOR STEP REJECTED) CRCT0135 CRCT0136 CRCT0137 С **CRCT0138 CRCT0140** END CRCT0141

54

T. Certa.

	SUBROUTINE TANGNT(NVAR;X;IP;TAN;IRET;ICALL;FPRYN;IPYT;NEQN;DETA; 1 [EXP]	TNGN0001 TNGN0002
C C #1	***************************************	TNGN0003
ç	CURRONTINE TANGUT COMPUTES THE HALT TANGENT HECTOR TO THE SOULTION	TNGN0005
č	CURVE OF THE UNDERDETERNINED NONLINEAR SYSTEM FX = 0. THE	TNGN0007
C C	TANGENT VECTOR TAN IS THE SOLUTION OF THE LINEAR SYSTEM	TNGN0008
Ĉ	RFA(X, IPL) * TAN = E(NVAR)	TNGN0010
č	WHERE DFA(X, IPL) IS THE NUAR BY NUAR MATRIX WHOSE FIRST NUAR-1 ROWS	STNGN0012
C C	ARE DFX/DX (X); THE DERIVATIVE OF FX EVALUATED AT X; AND WHOSE LAST ROW IS (E(IPL)) TRANSPOSE; THE NVAR COMPONENT EUCLIDEAN COORDINATE	TNGN0013 TNGN0014
ç	VECTOR WITH 1 IN THE IPL-TH POSITION AND ZEROS ELSEWHERE, E(NVAR) 19	STNGN0015
č	LAST COMPONENT.	TNGN0017
Č	THE TANGENT VECTOR IS THEN NORMALIZED AND ITS SIGN ADJUSTED,	TNGN0018
C C	INPUT NVAR = THE NUMBER OF VARIABLES	TNGN0020 TNGN0021
Ĉ	X = THE CURRENT CONTINUATION POINT	TNGN0022
Č	IF = CONTINUATION CONFUNENT SET UN LAST STEF	TNGN0023
C C	DRITPUT TAN = THE UNIT TANGENT VECTOR IN CONTINUATION DIRECTION AT X	TNGN0025 TNGN0026
Č	DETA = BINARY MANTISSA OF DETERMINANT OF JACOBIAN DFA(X, IPL)	TNGN0027
č	IP = LOCATION OF LARGEST COMPONENT OF TANGENT VECTOR TAN	TNGN0029
C C	CANDIDATE FOR NEW CONTINUATION COMPONENT	TNGN0030 TNGN0031
Ĉ	THIS SUBROUTINE IS CALLED BY	TNGN0032
č	AND CALLS	TNGN0034
C Ç	Solve LINPAK ISAMAX	TNGN0035 TNGN0036
C C	LINPAK SNRH2	TNGN0037
	EINTHA DOLAL	1 201500(0).426
Č,		TNGN0038
C C C C		TNGN0039 *TNGN0040 TNGN0041
C C C	EINTHN SSUNE ####################################	TNGN0039 TNGN0039 TNGN0040 TNGN0041 TNGN0042 TNGN0043
C C C C	EINTHK SSCHE ISTATTATTATTATTATTATTATTATTATTATTATTATTAT	TNGN0039 TNGN0040 TNGN0041 TNGN0042 TNGN0043 TNGN0043 TNGN0044 TNGN0044
C#	EINTHK SSCHE EINTHK SSCHE REAL X(NVAR), TAN(NVAR), FPRYN(NVAR, NVAR) INTEGER IPVT(NVAR) COMHON /COUNT1/ ICRSL, ITNSL, NSTCR, NCNCR, NTRCR, NLMCR, NLMRT COMHON /COUNT2/ IFEVAL, ISOLVE, NRED, NRDSUM, KN, KNSUM COMHON /CUIPUT/ IWRITE	TNGN0038 TNGN0040 TNGN0041 TNGN0042 TNGN0043 TNGN0044 TNGN0044 TNGN0044 TNGN0046
	EINTHK SSCHE ISTATTATTATTATTATTATTATTATTATTATTATTATTAT	1 MGN00.38 TNGN0039 *TNGN0040 TNGN0041 TNGN0042 TNGN0043 TNGN0044 TNGN0045 TNGN0046 TNGN0047 TNGN0048
CCC CCC CCC	EINTHK SSCHL REAL X(NVAR), TAN(NVAR), FPRYN(NVAR, NVAR) INTEGER IPVT(NVAR) COMHON /COUNT1/ ICRSL, ITNSL, NSTCR, NCNCR, NTRCR, NLMCR, NLMRT COMHON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRDSUM, KN, KNSUM COMHON /CHITPUT/ IWRITE COMPUTE TANGENT VECTOR DD 10 I=1, NEGN	1 1000038 TNGN0039 TNGN0040 TNGN0041 TNGN0043 TNGN0044 TNGN0045 TNGN0046 TNGN0046 TNGN0047 TNGN0048 TNGN0049 TNGN0050
	EINTHK SSCHE ISTATTATTATTATTATTATTATTATTATTATTATTATTAT	1000038 TNGN0039 *TNGN0040 TNGN0042 TNGN0042 TNGN0043 TNGN0044 TNGN0046 TNGN0046 TNGN0046 TNGN0048 TNGN0049 TNGN0050 TNGN0050
	EINTHK SSCHE REAL X(NVAR), TAN(NVAR), FPRYH(NVAR, NVAR) INTEGER IPVT(NVAR) COHHON /COUNT1/ ICRSL, ITNSL, NSTCR, NCNCR, NTRCR, NLHCR, NLHRT COHHON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRDSUH, KN, KNSUH COHHON /CUITPUT/ IWRITE COMPUTE TANGENT VECTOR DO 10 I=1, NEGN 10 TAN(I)=0.0 TAN(NVAR)=1.0 IERR=0	1 NGN0038 TNGN0039 * TNGN0040 TNGN0041 TNGN0042 TNGN0043 TNGN0044 TNGN0045 TNGN0046 TNGN0047 TNGN0047 TNGN0048 TNGN0048 TNGN0050 TNGN0051 TNGN0052 TNGN0053
	EINTHK SSCHL REAL X(NVAR), TAN(NVAR), FPRYH(NVAR, NVAR) INTEGER IPVT(NVAR) COHHON /COUNT1/ ICRSL, ITNSL, NSTCR, NCHCR, NTRCR, NLHCR, NLMRT COHHON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRDSLH, KN, KNSUH COHHON /CUITPUT/ IWRITE COMPUTE TANGENT VECTOR DO 10 I=1,NEGN 10 TAN(I)=0.0 TAN(NVAR)=1.0 IERR=0 CALL SOLVE(NVAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IHOD, FPRYH, 1 [PVT]	1 ngn00.38 TNGN0040 TNGN0040 TNGN0041 TNGN0042 TNGN0043 TNGN0044 TNGN0045 TNGN0046 TNGN0046 TNGN0047 TNGN0048 TNGN0050 TNGN0050 TNGN0053 TNGN0053 TNGN0055
	EINTHK SSCHE REAL X(NVAR), TAN(NVAR), FPRYH(NVAR, NVAR) INTEGER IPUT(NVAR) COMMON /COUNT1/ ICRSL, ITNSL, NSTCR, NCNCR, NTRCR, NLMCR, NLMRT COMMON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRDSLM, KN, KNSUM COMMON /OUTPUT/ IWRITE COMPUTE TANGENT VECTOR DO 10 I=1, NEGN 10 TAN(I)=0.0 TAN(NVAR)=1.0 IERR=0 CALL SOLVE(NVAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IMOD, FPRYM, 1 IPUT) ITNSL=ITNSL+1 ISOLVE(NVAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IMOD, FPRYM,	1 1000038 TNGN0039 *TNGN0040 TNGN0042 TNGN0042 TNGN0043 TNGN0044 TNGN0045 TNGN0045 TNGN0054 TNGN0055 TNGN0055 TNGN0055
	<pre>EINTHK SSCAL IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII</pre>	1 1000038 TNGN0039 TNGN0040 TNGN0041 TNGN0044 TNGN0044 TNGN0044 TNGN0046 TNGN0046 TNGN0046 TNGN0047 TNGN0047 TNGN0048 TNGN0051 TNGN0051 TNGN0051 TNGN0055 TNGN0055 TNGN0055 TNGN0057 TNGN0057 TNGN0057
	<pre>EINTHK SSCHE REAL X(NVAR), TAN(NVAR), FPRYH(NVAR, NVAR) INTEGER IPVT(NVAR) COHHON /COUNT1/ ICRSL, ITNSL, NSTCR, NCMCR, NTRCR, NLMCR, NLMRT COHHON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRDSUH, KN, KNSUH COHHON /CUITPUT/ IWRITE COMPUTE TANGENT VECTOR DO 10 I=1,NEGN 10 TAN(I)=0.0 TAN(NVAR)=1.0 IERR=0 CALL SOLVE(NVAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IHOD, FPRYH, 1 IPVT) ITNSL=ITNSL+1 ISOLVE=ISOLVE+1 IF (IERR.NE.0) IRET=-1 IF (IRET.LT.0) RETURN </pre>	1 1000038 TNGN0039 TNGN0040 TNGN0040 TNGN0042 TNGN0043 TNGN0045 TNGN0045 TNGN0046 TNGN0048 TNGN0048 TNGN0050 TNGN0050 TNGN0051 TNGN0055 TNGN0055 TNGN0056 TNGN0058 TNGN0058 TNGN0058 TNGN0058
000 000 000	REAL X(NVAR), TAN(NVAR), FPRYN(NVAR, NVAR) INTEGER IPVT(NVAR) COHHON /COUNT1/ ICRSL, ITNSL, NSTCR, NCNCR, NTRCR, NLMCR, NLMRT COHHON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRDSUM, KN, KNSUM COHHON /CUITPUT/ IWRITE COMPUTE TANGENT VECTOR 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, IHOD, FPRYN, 1 IPVT) ITNSL=ITNSL+1 ISOLVE=ISOLVE+1 IF (IERR.NE.0) IRET=-1 IF (IRET.LT.0) RETURN OBTAIN EUCLIDEAN NORM OF TANGENT VECTOR	1 1000038 TNGN0039 * TNGN0040 TNGN0041 TNGN0042 TNGN0044 TNGN0044 TNGN0045 TNGN0046 TNGN0050 TNGN0051 TNGN0052 TNGN0055 TNGN0056 TNGN0057 TNGN0057 TNGN0058 TNGN0058 TNGN0050 TNGN0050 TNGN0050 TNGN0056 TNGN0050 TNGN0050 TNGN0050 TNGN0052
	<pre>EINTHAL SOCHE EINTHAL EINTHEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE</pre>	1 1000039 TNGN0039 TNGN0040 TNGN0040 TNGN0041 TNGN0042 TNGN0044 TNGN0045 TNGN0045 TNGN0050 TNGN0050 TNGN0055
	<pre>ELAMPHK_SSCHE ELAMPHK_SSCHE REAL_X(NVAR),TAN(NVAR),FPRYH(NVAR,NVAR) INTEGER IPUT(NVAR) COMMON /COUNT1/ ICRSL,ITMSL,NSTCR,NCNCR,NTRCR,NLHCR,NLHRT COMMON /COUNT2/ IFEVAL,ISOLVE,NRED,NRDSUM,KN,KNSUM COMMON /OUTPUT/ IMRITE COMPUTE TANGENT VECTOR DO 10 I=1,NEGN 10 TAN(I)=0.0 TAN(NVAR)=1.0 IERR=0 CALL_SOLVE(NVAR,X,TAN,IP,DETA,IEXP,IERR,ICALL,IMOD,FPRYM, 1 IPUT) ITMSL=ITMSL+1 ISOLVE=ISOLVE+1 IF (IRET.LT.0) RETURN OBTAIN EUCLIDEAN NORM OF TANGENT VECTOR IP=ISAMAX(NVAR,TAN,1) IF (INORM-ENRM2(NVAR,TAN,1)) IF (INORM.ER.0.0) IRET=-2 YE (IDET.LT.0) GETURN IF (INORM.ER.0.0) IF</pre>	1 1000038 TNGN0039 * TNGN0040 TNGN0042 TNGN0042 TNGN0043 TNGN0044 TNGN0045 TNGN0046 TNGN0047 TNGN0047 TNGN0047 TNGN0047 TNGN0052 TNGN0052 TNGN0055 TNGN055 TNG
	REAL X(NVAR), TAN(NVAR), FPRYH(NVAR, NVAR) INTEGER IPVT(NVAR) COMMON /COUNT1/ ICRSL, ITNSL, NSTCR, NCNCR, NTRCR, NLMCR, NLMRT COMMON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRBSUN, KN, KNSUM COMMON /CUINT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRBSUN, KN, KNSUM COMMON /CUITPUT/ IWRITE COMPUTE TANGENT VECTOR DO 10 I=1, NEON 10 TAN(I)=0.0 TAN(I)=0.0 TAN(I)=0.0 CALL SOLVE(NVAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IMOD, FPRYN, 1 IPVT) ITNSL=ITNSL+1 ISOLVE=ISOLVE+1 IF (IERR.NE.0) IRET=-1 IF (IERT.LT.0) RETURN OBTAIN EUCLIDEAN NORM OF TANGENT VECTOR IP=ISAMAX(NVAR, TAN, 1) IF (TNORM=SNRM2(NVAR, TAN, 1) IF (INCH, ER.0.0) IRET=-2 IF (IRET.LT.0) RETURN	1 MGN0038 TNGN0039 TNGN0040 TNGN0040 TNGN0044 TNGN0044 TNGN0044 TNGN0046 TNGN0046 TNGN0047 TNGN0047 TNGN0049 TNGN0050 TNGN0050 TNGN0051 TNGN0055 TNGN0055 TNGN0057 TNGN0057 TNGN0057 TNGN0057 TNGN0057 TNGN0056 TNGN0060 TNGN0064 TNGN0065 TNGN0065
	REAL X(NVAR), TAN(NVAR), FPRYN(NVAR, NVAR) INTEGER IPUT(NVAR) COMMON /COUNT1/ ICRSL, ITNSL, NSTCR, MCMCR, NTRCR, NLMCR, NLMRT COMMON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRDSUM, KN, KNSUM COMMON /COUTPUT/ IWRITE COMPUTE TANGENT VECTOR DO 10 I=1, NEGN 10 TAN(I)=0.0 TAN(NVAR)=1.0 IERR=0 CALL SOLVE(NVAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IMOD, FPRYM, 1 IPUT) ITNSL=ITNSL+1 ISOLVE=ISOLVE+1 IF (IERR, ME.O) IRET=-1 IF (IERT, LT.0) RETURN OBTAIN EUCLIDEAN NORM OF TANGENT VECTOR IP=ISAMAX(NVAR, TAN, 1) INORM=SNRM2(NVAR, TAN, 1) IF (TNORM, ER.0.0) IRET=-2 IF (IRET, LT.0) RETURN NORMALIZE THE VECTOR	1 1000039 TNGN0039 TNGN0040 TNGN0040 TNGN0042 TNGN0042 TNGN0044 TNGN0045 TNGN0045 TNGN0046 TNGN0050 TNGN0050 TNGN0055 TNGN0055 TNGN0055 TNGN0055 TNGN0055 TNGN0057 TNGN0057 TNGN0057 TNGN0055 TNGN0055 TNGN0055 TNGN0055 TNGN0065 TNGN0064 TNGN0065 TNGN0065 TNGN0065 TNGN0065 TNGN0065 TNGN0065
	REAL X(NUAR), TAN(NUAR), FPRYH(NUAR, NUAR) INTEGER IPUT(NUAR) COHHON /COUNT2/ ICRSL, ITNSL, NSTCR, NCNCR, NTRCR, NLHCR, NLHRT COHHON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRDSLM, KN, KNSUM COHHON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRDSLM, KN, KNSUM COHHON /COUTUT/ IWRITE COMPUTE TANGENT VECTOR DO 10 I=1, NEGN 10 TAN(I)=0.0 TAN(NUAR)=1.0 IERR=0 CALL SOLVE(NVAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IHOD, FPRYH, 1 IFUT) ITNSL=ITNSL+1 ISOLVE(NVAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IHOD, FPRYH, 1 IFUT) ITNSL=ISOLUE+1 IF (IERT, NE.0) IRET=-1 IF (IRET, LT.0) RETURN OBTAIN EUCLIDEAN NORM OF TANGENT VECTOR IP=ISAMAX(NUAR, TAN, 1) INORH=SNRM2(NVAR, TAN, 1) IF (INCRER.0.0) IRET=-2 IF (IRET, LT.0) RETURN NORNALIZE THE VECTOR SCALER=1.0/TNORM CALL SECAL (NUAR, SCALER-TAN, 1)	1 1000039 TNGN0039 TNGN0040 TNGN0040 TNGN0042 TNGN0044 TNGN0044 TNGN0045 TNGN0046 TNGN0045 TNGN0050 TNGN0050 TNGN0055 TNGN0056 TNGN0056 TNGN0056 TNGN0057 TNGN0056 TNGN0065 TNGN0064 TNGN0065 TNGN0055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TNGN055 TN
	REAL X(NVAR), TAN(NVAR), FPRYN(NVAR, NVAR) INTEGER IPVT(NVAR) COMMON /COUNT1/ ICRSL, ITWSL, NSTCR, NCHCR, NTRCR, NLMCR, NLMRT COMMON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRDSUM, KN, KNSUM COMMON /COUNT2/ IWRITE COMPUTE TANGENT VECTOR DO 10 I=1,NEON 10 TAN(I)=0.0 TAN(NVAR)=1.0 IERR=0 CALL SOLVE(NVAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IMOD, FPRYN, 1 IPUT) ITMSL=ITNSL+1 ISOLVE=1SOLVE+1 IF (IERR.NE.0) IRET=-1 IF (IRET.LT.0) RETURN OBTAIN EUCLIDEAN NORM OF TANGENT VECTOR IP=ISAMAX(NVAR, TAN, 1) IF (TNORM-ER.0.0) IRET=-2 IF (IRET.LT.0) RETURN NORMALIZE THE VECTOR SCALER=1.0/TNORM CALL SSCAL(NVAR, SCALER, TAN, 1) RETURN	1 1000039 TNGN0039 TNGN0040 TNGN0040 TNGN0044 TNGN0044 TNGN0044 TNGN0045 TNGN0046 TNGN0050 TNGN0050 TNGN0050 TNGN0055 TNGN0050 TNGN0055 TNGN055 TN
	<pre>ELMCHAL SOLAL REAL X(NUAR), TAN(NUAR), FPRYH(NUAR, NUAR) INTEGER IPUT(NUAR), COMMON /COUNTL/ ICRSL, IINSL, NSTCR, NCHCR, NTRCR, MLHCR, NLHRT COMMOM /COUNTL/ ICRSL, IINSL, NSTCR, NCHCR, NTRCR, MLHCR, NLHRT COMMOM /COUTPUT/ IWRITE COMPUTE TANGENT VECTOR DD 10 I=1.NEGN 10 TAN(I)=0.0 TAN(MUAR)=1.0 IERR=0 CALL SOLVE(NUAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IMOD, FPRYM, 1 IPUT) ITMSL=ITNSL+1 ISOLVE[NUAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IMOD, FPRYM, 1 IPUT) ITMSL=TINSL+1 IF (IERR.HC.0) IRET=-1 IF (IERR.HC.0) IRET=-2 IF (IRRT.LT.0) RETURN OBTAIN EUCLIDEAN NORM OF TANGENT VECTOR IP=ISAMAX(NUAR, TAN, 1) IF (TMORM-ER.0.0) IRET=-2 IF (IRET.LT.0) RETURN NORNALIZE THE VECTOR SCALFR=1.0/TNORM CALL SSCAL(NUAR, SCALER, TAN, 1) RETURN ***********************************</pre>	1 116900.38 TNGN0039 *TNGN0040 TNGN0041 TNGN0042 TNGN0042 TNGN0044 TNGN0045 TNGN0045 TNGN0045 TNGN0050 TNGN0051 TNGN0052 TNGN0055 TNGN0055 TNGN0055 TNGN0055 TNGN0055 TNGN0055 TNGN0055 TNGN0055 TNGN0055 TNGN0055 TNGN0061 TNGN0064 TNGN0065 TNGN0070 TNGN0071 TNGN0073 TNGN0073
	EINTHA SSUME EINTHA SSUME REAL X(NVAR), TAN(NVAR), FPRYN(NVAR, NVAR) INTEGER IPUT(NVAR) COMMON /COUNT1/ ICRSL, ITNSL, NSTCR, NCMCR, NTRCR, NLMCR, NLMRT COMMON /COUNT2/ IFEVAL, IPEVAL, ISOLVE, NRED, NRBUH, KN, KNSUM COMMON /OUTPUT/ IWRITE COMPUTE TANGENT VECTOR DO 10 I=1.NEGN 10 TAN(I)=0.0 TAN(MVAR)=1.0 IERR=0 CALL SOLVE(NVAR, X, TAN, IP, DETA, IEXP, IERR, ICALL, IMOD, FPRYN, 1 IPUT) ITMSL=ITNSL+1 ISOLVE=ISOLVE+1 IF (IRET.LT.0) RETURN OBTAIN EUCLIDEAN NORM OF TANGENT VECTOR IP=ISAMAX(NVAR, TAN, 1) TMORM=SNRM2(NVAR, TAN, 1) IF (TNORM, EQ.0.0) IRET=-2 IF (IRET.LT.0) RETURN MORNALIZE THE VECTOR SCALER=1.0/TNORM CALL SSCAL(NVAR, SCALER, TAN, 1) RETURN FND	1 1000039 TNGN0039 TNGN0040 TNGN0040 TNGN0041 TNGN0044 TNGN0044 TNGN0044 TNGN0045 TNGN0050 TNGN0050 TNGN0055 TNGN055 TNGN0

.

	SHR	ROUTINE ROOT(A.FA.R.FR.C.FC.KOUNT.IFLAG)	R0010001
C	0.764		R0010002
Ū‡1	******	***************************************	ROOTOOO3
C			R00T0004
č	SUBROU	TINE ROOT SEEKS A ROOT OF THE EQUATION F(X)=0.0,	ROOT0005
C	ULVEN I	A STAKLING INTERVAL (A)C) UN WHICH F CHANDES SIGN,	R0010006
L C	CA AND	ST LALL TU KUUTT THE INTERVAL AND FUNCTION VALUES. Se add fer in and an approvingtion r for the post is settimate	R0010007
ř	REFURE	FOR SUBSEDIENT CALL, THE USED EVALUATES FREE(R), AND THE	POOTOOOS
č	PROGRA	N TRIES TO RETURN A BETTER APPROXIMATION 8.	ROOT0010
č			R00T0011
Ĉ	THIS P	ROGRAM IS BASED ON THE FORTRAN FUNCTION ZERO	R00T0012
Ç	GIVEN	IN THE BOOK:	R00T0013
Ç	'AL GOR	ITHMS FOR MINIMIZATION WITHOUT DERIVATIVES'	RODTOO14
Č	BY RIC	HARD P. BRENT, PRENTICE HALL, INC, 1973	R0010015
۲ ۲		NICICATIONS USED DONE BY JOHN DIDKADDT	RUUT 0016
ř	INE NU	DIFICHTIOND WERE DURE BI JUDH DURNMRDI+	EDOTOO18
č		NT:	R0010019
č			R00T0020
Ċ	A	- IS ONE ENDPOINT OF AN INTERVAL IN WHICH F CHANGES SIGN.	R00T0021
С	FA	- THE VALUE OF F(A). THE USER HUST EVALUATE F(A) BEFORE FIRST	R00T0022
Ç		CALL ONLY. THEREAFTER THE PROGRAM SETS FA.	R00T0023
Č	B	- ON FIRST CALL, B SHOULD NOT BE SET BY THE USER,	R0010024
L C		UN SUNSEAUENT GALLST K SMUULI MUT BE GARANDED	RUU10023
ř		TRUTIIS OUTPUT VALUET THE GURKENT AFTRUALHAN	RUU19926
č	FR	- IN FIRST CALL, FR SHALL D NOT RE SET BY THE USER.	80010028
č		THEREAFTER, THE USER SHOULD EVALUATE THE FUNCTION	R00T0029
č		AT THE OUTPUT VALUE B, AND RETURN FR=F(B).	R00T0030
Ĉ	C	- IS THE OTHER ENDPOINT OF THE INTERVAL IN WHICH	R00T0031
Ç		F CHANGES SIGN. NOTE THAT THE PROGRAM WILL RETURN	R00T0032
Č	~~	INNEDIATELY WITH AN ERROR FLAG IF FC*FA.GT.0.0.	ROUTOO33
ç	FC	- THE VALUE OF F(C). THE USER MUST EVALUATE F(C) REFORE FIRST	RUU100.34
۲ ۲	KOUNT	- A COUNTER FOR THE NUMBER OF CALLS TO POOT. KOUNT	RUGIOUSS ROOTOOSS
č	1004	SHOWN RE SET TO TERO ON THE FIRST CALL FOR A GIVEN	R0010037
č		ROOT PROBLEM.	R00T0038
č	IFLAG	- AN ERROR RETURN FLAG WHOSE INPUT VALUE IS INHATERIAL.	R00T0039
Ĉ			R00T0040
Č	ON RET	URN FROM A CALL TO ROOT	R0010041
C			RODT0042
L C	H FA	- UNE ENDPOINT OF CHANGE OF SIGN INTERVAL+	RUUI 0043
č	8	- CURRENT APPROXIMATION IN THE ROOT. REFORE ANOTHER	ROOTOOAS
č	*	CALL TO ROOT, EVALUATE F(B).	R0010046
С	FB	- FB WILL BE OVERWRITTEN BY THE USER BEFORE ANOTHER	RODT0047
C		CALL, ITS VALUE ON RETURN IS ONE OF FA, FR OR FC.	R00T0048
Č	Č_	- OTHER ENDPOINT OF CHANGE IN SIGN INTERVAL.	R00T0049
C	FC	- THE VALUE OF F(C).	R00T0050
č	KOUNT	- CURRENT NUMBER OF CALLS TO ROOT.	RODT0051
Ľ	11100	TELACE O MEANO THAT ON STRET CALL, EATER OT A A	NU010052
ř		THIS IS AN EDDAD RETIRD. SINCE A REACKETING	
ř		INTERUAL SHOULD RE SUPPLIED ON FIRST CALL.	R0010055
č		IFLAG=-1 NEANS THAT THE CURRENT BRACKETING INTERVAL	R0010056
Ē		WHOSE ENDPOINTS ARE STORED IN A AND C	R00T0057
С		IS SO SHALL (LESS THAN 4*EPHACH*ABS(B) +EPHACH)	R00T0058
Č		THAT B SHOULD BE ACCEPTED AS THE ROOT.	ROOT0059
č		THE FUNCTION VALUE F(B) IS STORED IN FB.	RUUTO060
2		ITLAUT V ARAAD IANI IAR IARUI VALUR PE ID RAAVILI 7600. And D Choin B de accedted ac the doot	RUUI VV01
č		LERUT MAN D SHUULD BE MUULFILD MO INE RUUI.	RUU I VVG2 RANTAA27
č		IFLAG.GT.O NEANS THAT THE CURRENT APPROXIMATION TO	RINTOOA
č		THE ROOT IS CONTAINED IN B. IF A BETTER	ROOTOOAS
Ē		APPROXIMATION IS DESIRED, SET FR=F(R)	RODTOOA6
Ç		AND CALL ROOT AGAIN. THE VALUE OF IFLAG INDICATES	R00T0067
C		THE METHOD THAT WAS USED TO PRODUCE B.	RDDT0068
č			RUDT0069
U		APLAGE & BASECIAUN WAS USEN.	KQU10070

	•	
č	IFLAG= 2 LINEAR INTERPOLATION (SECANT METHOD).	R0010071
с С	THE 2 THAT SE ADADKATTE THE KAATTAL	R0010072
ē.	LOCAL VARIABLES INCLUDE:	R00T0074
Č		ROOT0075
C C	EPRACH- SMALLEST PUSITIVE NUMBER SUCH THAT 1.01EPRACH.61.1.0 SUBSTAUM(1-TAN) FOR POWNER, TAN-DIGIT APTIMETIC	R00100/6
č	RASE RETA. THICE THAT VALUE FOR TRUNCATED ARTHMETIC.	R0010078
č	THIS IS THE RELATIVE MACHINE PRECISION.	R0010079
Č	HALFBC- SIGNED HALFWIDTH OF INTERVAL, DURING SEGNENT 3, THE	R00T0080
č	CHANGE OF SIGN INTERVAL IS (B)C) OR (C)R), THE MIDPOINT	RUGT0081
č	SNELL - STYF OF CHANGE IN STRN INTERVAL .	R0010082
č	SDEL2 - PREVIOUS VALUE OF SDEL1.	R0010084
Ç	SDEL3 - PREVIOUS VALUE OF SDEL2.	R00T0085
č	SDEL 4 - PREVIOUS VALUE OF SDEL3.	R0010086
r r	SIEP - THE NEW KUUT IS CONFULLY AS A COKKELITON TO A UP THE.	ROOTOOSA
č	TOLER - A NUMBER WE ACCEPT AS 'SHALL' WHEN EXAMINING INTERVAL	R00T0089
Ĉ	SIZE OR STEP SIZE. TOLER=2.0*EPMACH*ABS(B) + EPMACH IS	R00T0090
Č	A MINIMUM BELOW WHICH WE WILL NOT ALLOW SUCH VALUES TO FALL.	RODT0091
C C	INIS SUBRUULINE, IS CALLED BT	R0010072
č	AND CALLS	R0010094
č	FORTRAN ABS	R00T0095
Ç	FORTRAN SIGN	R0010096
		KUUIIV09/
С. С.	** ******************************** ******	R0010078
•	REAL A, B, C, FA, FR, FC, STEP, TOLER, P, Q, R, S	ROUTOLOO
	CONNON /TOL/ EPHACH, EPSATE, EPSQRT	R00T0101
C	CECMENT 11 ETDET CALL MANDLED CDEPTALLY DD DOOKYEEDING	RU010102
č	SCORENT IF FINST UNLL NONDLED SFECIPLETE DU BUBANCEFINGT	R0010104
Ē.	SET CERTAIN VALUES ONLY FOR INITIAL CALL WITH KOUNT=0	R00T0105
C		R00T0106
	IF (KOUNT,GT,O) GO TO 10 TE (EA.GT.O.O.AND.EC.ST.O.O) GO TO 110	RU010107
	IF (FA.LT.O.O.ANB.FC.LT.O.O) 60 TO 110	ROOT0109
	KOUNT=1	R00T0110
	SDEL1=2.0#ABS(C-A)	R00T0111
	51R.L2=2+0=51R.L1 SDEL3=2-0=8DEL2	R0019112
	Bac Bac	ROOTO114
	FB=FC	R00T0115
~	GQ TQ 20	R00T0116
č	ON SUERY CALL. INCREMENT COUNTER	R0019117
č		R00T0119
-	10 KOUNT=KOUNT+1	R0010120
č	DETINN TE UTT MACUTHE TEDO FOD F/D	R00T0121
C C	RETURN IF HIT HAGHINE LERU FOR F(B)	ROUT0122
	IF(FR.EQ.0.0) 60 TO 90	R00T0124
C		R00T0125
Č	SEGNENT 2: REARRANGE POINTS AND FUNCTION VALUES IF	R0010126
C	ARGERSART SU THAT FRATCALIAUAU AND SU THAT ARGERIALTARS(FC)	R0010127
č		R00T0129
-	IF((FR.LE.0.0).AND.(FC.8T.0.0)) 60 TO 30	ROOTO130
~	IF((FB.GT.0.0).AND.(FC.LE.0.0)) GD TO 30	R00T0131
č	FR AND FC ARE OF SAME SIGN.	R0010132
č	(ROOT CHANGED SIGN)	R00T0134
Ç	OVERWRITE C WITH VALUE OF A	ROOT0135
C	20 F=A	RUUT0136
		R0010137

C IF NECESSARY, SET A:=R, B:=C, C:=B TO ENSURE THAT ABS(FB).LE.ABS(FC) Ĉ Ċ 30 IF(ARS(FC).GE.ABS(FR)) 60 TO 40 A=Ŗ R=C C=A FA=FB FR=FC FC=FA C SEGNENT 3: CHECK FOR ACCEPTANCE BECAUSE OF SHALL INTERVAL CURRENT CHANGE IN SIGN INTERVAL IS (C+B) OR (B+C). Ĉ C 40 TOLER=2.0*EPHACH*ABS(B)+EPHACH HALFRC=0.5#(C-B) SDEL4=SDEL3 SDEL 3=SDEL2 SDEL2=SDEL1 SDEL1=ABS(C-B) IF (ABS (HALFBC) .LE. TOLER) GO TO 100 SEGNENT 4: COMPUTE NEW APPROXIMANT TO ROOT OF THE FORM C BINEW)=B(OLD)+STEP. METHODS AVAILABLE ARE LINEAR INTERPOLATION INVERSE QUADRATIC INTERPOLATION C Č C С AND BISECTION. C IF(ARS(FR).GE.ARS(FA))60 TO 70 IF(A.NE.C) 60 TO 50 ATTEMPT LINEAR INTERPOLATION IF ONLY TWO POINTS AVAILABLE COMPUTE P AND Q FOR APPROXIMATION B(NEW)=B(OLD)+P/QĈ Ĉ IFLAG=2 S=FB/FA P=2.0#HALFBC#S Q=1.0-5 60 TO 60 CCC ATTEMPT INVERSE QUADRATIC INTERPOLATION IF THREE POINTS AVAILABLE COMPUTE P AND R FOR APPROXIMATION B(NEW) = B(OLD) + P/RC 50 JFLAG=3 S=FB/FA Q=FA/FC R=FR/FC P=5#(2.0#HALFBC#Q#(R-R)-(B-A)#(R-1.0)) g=(g-1.0)#(R-1.0)#(S-1.0) £ CORRECT THE SIGNS OF P AND Q C C 60 IF(P.GT.0.0)Q=-Q P=ABS(P) С C IF P/9 IS TOO LARGE, GO BACK TO BISECTION IF(8.0*SDEL1.GT.SDEL4) GO TO 70 IF (P.GE.1.5#ABS(HALFBC#R)-ABS(TOLER#R)) GO TO 70 STEP=P/Q GO TO 80 Č PERFORM BISECTION: TF ABS(FB).GE.ABS(FA) OR INTERPOLATION IS UNSAFE (P/Q IS LARGE) OR IF THREE CONSECUTIVE STEPS H/VE NOT DECREASED THE SIZE OF THE INTERVAL BY A FACTOR OF 8.0 Č c c 70 IFLAG=1 STEP=HALFRC GO TO 80

58

R00T0139

R00T0140

R00T0141

R00T0142

R00T0143

R0010144

R0010145

R00T0146

R00T0147

R00T0148

R00T0149

R00T0150

R00T0151 R00T0152

R00T0153

R00T0154

R00T0155 R00T0156

RDOT0157

R00T0158

R00T0159

R00T0160 R00T0161

R00T0162

R00T0163

R00T0164

R00T0165

R00T0166

R00T0167

R00T0168 R00T0169 R00T0170

R00T0171

R00T0173

R00T0174

R00T0175 R00T0176

R00T0177

R00T0178 R00T0179

R0010180 R0010181 R0010182

R00T0183 R00T0184

R00T0185

R00T0186

R00T0187 R00T0188

R00T0189

R00T0190

R00T0191

R00T0192

R00T0193

R00T0194

R00T0195 R00T0196

R0010197 R0010198

R00T0199

R00T0200 R00T0201

R00T0202

ROOT0203 ROOT0204

R0010205 R0010206 R0010207

ROOTOZOR

ROOTO209

R00T0210

R00T0211 R00T0212 00000 SEGNENT 5: VALUE OF STEP HAS REFN COMPUTED. UPDATE INFORMATION: A:=B, FA:=FB, B:=B+STEP. R00T0213 CHANGE IN SIGN INTERVAL IS NOW (A,C) OR (C,A). R00T0214 R00T0215 R00T0216 80 A=8 R00T0217 FA=FB R00T0218 IF(ABS(STEP).LE.TOLER) STEP=SIGN(TOLER, HALFRC) R00T0219 R00T0220 B=B+STEP RETURN R00T0221 R00T0222 CCCCCC SPECIAL RETURNS R00T0223 R00T0224 INPUT POINT & IS EXACT ROOT R00T0225 R00T0226 90 IFL.AG=0 RIATIO223 ROOT0227 ROOT0228 ROOT0229 ROOT0230 ROOT0231 ROOT0232 RETURN C CHANGE IN SIGN INTERVAL IS OF SIZE LESS THAN 4*EPMACH*ABS(B)+EPMACH INTERVAL RETURNED AS (B,C) OR (C,B). ACCEPT B AS ROOT WITH RESIDUAL F(B) STORED IN FB. CCCC R0010232 R0010233 R0010234 R0010235 R0010236 R0010237 100 IFLAG=-1 A=B FA=FB RETURN C R0010239 R0010239 R0010240 R0010241 Ĉ CHANGE OF SIGN CONDITION VIOLATED C 110 IFLAG=-2 KOUNT=0 RETURN R00T0242 ٢ R00T0245 END R00T0246

and a second

	SUR	ROUTINE_SOLVE(NVAR;X;Y;IP;DETA;IEXP;IERR;ICALL;IMOD;FPRYM; J])	SLVE0001
C			SLVE0003
Č*1	******	L*************************************	KSL VEOOO4
ř		INDRUITINE TO CALLED BY	SLVEOOOS
č	CORF	T	SEVENDOS
č	TANG	ίτ.	SLVE0008
Ĉ.	AND CA	LS	SI.VE0009
Ç	FORT	RAN_ABS	SLVE0010
Č	LINP	AK SGEFA	SL.VE0011
C C	LUNP	NA SGESL	SLVE0012
с r	UDER	FFKINE	SEVENOIS
č	THIS S	IBROUTINE SOLVES THE LINEAR SYSTEM DEA(X, IP) #YOUT = YIN	SI VE0015
Ĉ	WHERE	DFA(X, JP) IS THE (NVAR)X(NVAR) MATRIX WHOSE FIRST NVAR - 1	SLVE0016
C	ROWS A	RE THE JACOBIAN COMPUTED BY FPRIME, AND WHOSE LAST	SLVE0017
č	ROW IS	ALL O EXCEPT FOR A 1 IN THE IP-TH COMPONENT.	SLVE0018
с Г	VTN TO	THE MUAD COMPONENT HERTOD Y ON THOUT, AND THE COMMITTON	SLUEDODO
č	VECTOR	YNIT IS RETIRNED IN Y ON OUTPUT AFTER A SUCCESSED	SEVENUE
č	SETUP	AND SOLUTION.	SLVE0022
Ċ			SL.VE.0023
Ç	**NOTE	** SURROUTINE SOLVE USES FULL MATRIX STORAGE TO SOLVE THE	SLVE0024
ក្ត	LINEAR	SYSTEN. THE USER MAY WISH TO REPLACE THIS ROUTINE WITH	SLVE0025
ř	UME. FRU	KE BUJIRU IU MIB FKURLEN.	SLVE0020
č	RETA	RINARY MANTISSA OF THE DETERMINANT OF JACORIAN DEA(X.IP)	SI UE0028
č	IEXP	BINARY EXPONENT OF THE DETERMINANT OF JACOBIAN DFA(X, IP)	SLVE0029
Ĉ	THOD	NEWTON METHOD FLAG.	SL. VE0030
č		INOD=0, JACOBJAN IS TO BE EVALUATED FOR EVERY CORRECTOR STEP	SLVE0031
č		AND EVERY TANGENT CALCULATION	SLVE0032
с Г		INUNEST JACUSIAN 15 10 KE EVALUATED UNLI FUK FIKST CUKKECTUK STED, AND EUEDY TANGENT CALCULATION	SLVEOUSS
č	TCALL	SET UP FLAG.	SL VE0035
Č		IF (ICALL.EQ.Q.AND.INOD.NE.O) DON'T RE-EVALUATED JACORIAN	SLVE0036
C	INFO	OUTPUT FROM SGEFA. IF INFO.NE.O, SGEFA FOUND A 7ERO	SLVE0037
Ğ		PIVOT WHEN ELIMINATING INFO-TH VARIABLE.	SLVE0038
Č	LEKK	REJUKN FLAGY O MEANS SUCCESSFUL SULUTIONY I MEANS FAILURE	SLVEQ039
č	Y	THE POINT AT UNITH TO CUALLATE CORVE	SLVEUU4U
č	Ŷ	THE RIGHT HAND SIDE ON INPUT, THE SOLUTION	SLUE0042
Ĉ	•	ON OUTPUT	SLVE0043
C	FPRYM	ARRAY WHERE DFA(X, IP) IS TO BE STORED.	SLVE0044
č	IPVT	INTEGER WORK SPACE FOR PIVOT ROW SWITCHES DEMANDED BY SGEFA	SLVE0045
с r	11	THE VARIABLE USED IN THE AUDIENTIANS EQUALTON THAT IS UP THE	SLVE9046
č		TERN FACEPT FOR A 1.0 IN THE TRATH COLUMN.	
č			SLUE0049
Ċ\$	******	***************************************	*SLVE0050
C	DEA		SLVE0051
	KE.A	L X(NVAK))T(NVAK))TPKTA(NVAK)NVAK) EGED (PHT/MHAD)	SLVE0052
	COM	HON /COUNT2/ IFFUAL · IPFUAL · ISOL VE · NRED · NRDSUM · KN · KNSUM	SLUE0054
C			SL.VE0055
C	DEPEND	ING ON VALUES OF ICALL AND INOD, EITHER SET UP	SLVE0056
C C	AUGMEN	TED JACUBIAN, DECOMPOSE INTO L-U FACTORS, AND GET DETERMINANT,	SLVE0057
с Г	UK USE	CONTENT PACIFICED JACOPIAN WITH NEW KIGHT MANU SILL.	SLVEUUDB
-	IF	(ICALL.ER.0.AND.IMOD.NE.0) 60 TO 50	SLVEDUAT
	CAL	FPRIME (NVAR, X, FPRYM)	SLVE0061
	IPE	VAL=JPEVAL+1	SI.VE0062
	DO :	10 I=1,NVAR	SL.VE0063
	10	*PRTM(NVAR)[)=0.0 */*/*********************************	SLVE0064
r	PPX	TR(RVAK)/F)=1.0	SUVEOUGO
č	CARRY	OUT IN CORE LU DECOMPOSITION OF NUAR BY NUAR MATRIX	SLVE0067
Ĉ	AND US	E PIVOT INFORMATION TO COMPUTE DETERMINANT	SLVE0068

C	CALL_SGEFA(FPRYH+NVAR+NVAR+IPVT+INF0) DETA=1.0 IEXP=0	SLVE0069 SLVE0070 SLVE0071 SLVE0072
	TW0=2.0	SL.VE0073
	DU 40 (F)JRVAR Te (TDNT(T) NE T) DETA-DETA	SLVE0074
	AF (177()), NE, A) DE N==DE N NE TA=EDPVM(T, 1) *NE TA	SLVEU073
	T = (T = T + T + T + T + T + T + T + T + T +	SLUEA077
	20 IF $(ABS(DETA), GE, 1, 0)$ GO TO 30	SLUE0078
	DETA=DETA*THO	St UF0079
	IEXP=IEXP-1	SLVE0080
	60 TO 20	SL.VE0081
	30 IF (ABS(DETA).LT.TWO) GO TO 40	SLVE0082
	<u>DETA=DETA/THO</u>	SL.VE0083
	IEXP=IEXP+1	SLVE0084
		SLVE0085
		5L.VE.0086
c	IF (JAFULAELU) BU TU AU	
ř	HETWE I HE FACTORED MATRIX, COLUE CYCTEM HETWE EDGUARD-BARKHAOD	SUNEVIOR
č	STATUTION. AND OUF SHART FRIAN HAND STAF WITH SAULTION	SI VEOOR
č	ELEVELANT, THE DEMONTANT, MANY MARK WATH DEMONT	SLVE0091
•	50 CALL SGESL (FPRYH: NUAR . NUAR . IPUT. Y, 0)	SLVE0092
	IERR=0	SLVE0093
	RÊTÛRN	SLVE0094
	60 IERR=1	SL.VE0095
	INFO=0	SLVE0096
~	RETURN	SL.VE.0097
ι. Γ.		SLVE0098
υ#4 Γ	***************************************	SLVE9977
•	END	SLUEA1A1
		OF AE'OY AT

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.
- 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.
- 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.
- 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.
- 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 Lősung 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. Ponisch 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.
- 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 C²-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 C²-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.

