ALGORITHM 596
A Program for a Locally Parameterized Continuation Process

WERNER C. RHEINBOLDT and JOHN V. BURKARDT
University of Pittsburgh

Categories and Subject Descriptors. G.1. 5 [**Numerical Analysis**]: Roots of Nonlinear Equations—*iterative methods, systems of equations*; G.M [**Mathematics of Computing**]· Miscellaneous—*FORTRAN*

General Terms· Algorithms, Design

Additional Key Words and Phrases: Equilibrium problems, underdetermined systems of equations, solution manifolds of parameterized equations, continuation methods, local parameterization, limit point computation

1. DESCRIPTION

Let $F: R^n \to R^{n-1}$, $n \geq 2$, be a given, continuously differentiable mapping for which the regularity set

$$\mathscr{R}(F) = \{x \in R^n; \text{rank } DF(x) = n - 1\} \tag{1.1}$$

is nonempty. Moreover, suppose that the (underdetermined) system of $(n - 1)$ equations in n unknowns,

$$Fx = 0, \tag{1.2}$$

has at least one solution $x^0 \in \mathscr{R}(F)$. Then the regular solution set

$$\mathscr{E}_R(F) = \{x \in \mathscr{R}(F); Fx = 0\} \tag{1.3}$$

is an open, one-dimensional C^1-manifold in R^n. We are interested in computing the connected component $\mathscr{E}_R(F, x^0)$ of $\mathscr{E}_R(F)$ which contains x^0. By a fundamental result of differential geometry (see, e.g., [5]), $\mathscr{E}_R(F, x^0)$ is diffeomorphic either to a circle or to some interval (connected subset) of R^1. For simplicity we call $\mathscr{E}_R(F, x^0)$ the solution curve of (1.2) through x^0.

Received 28 November 1981, revised 22 July 1982, accepted 16 October 1982
This work was supported in part by National Science Foundation Grant MCS-78-05299 and in part by the Office of Naval Research under Contract N00014-77-C-0623.
Authors' address· Institute for Computational Mathematics and Applications, Department of Mathematics and Statistics, University of Pittsburgh, Pittsburgh, PA 15261.

The design of an algorithm for computing a sequence of points along this solution curve has been described in [3], [6], [7], and [8]. We sketch here only the general outline of the procedure.

An important role in the algorithm is played by the augmented mappings $F[i]$: $R^n \rightarrow R^n$, $1 \leq i \leq n$, defined by

$$F[i]x = \begin{pmatrix} Fx \\ (e^i)^{\mathrm{T}}x \end{pmatrix}, \qquad \forall x \in R^n, \tag{1.4}$$

where $e^1, \ldots, e^n$ denote the natural basis vectors of R^n. Since for $x \in \mathscr{R}(F)$ the $(n-1) \times n$ Jacobian matrix $DF(x)$ has rank $n-1$, there exist indices i, $1 \leq i \leq n$, such that the matrix

$$DF[i](x) = \begin{pmatrix} DF(x) \\ (e^i)^{\mathrm{T}} \end{pmatrix} \tag{1.5}$$

is nonsingular.

A first use of the augmented operators is in the computation of the tangent direction at any $x \in \mathscr{R}(F)$. More specifically, if i is such that (1.5) is nonsingular, then the tangent vector Tx is uniquely defined by the generic algorithm

$$\begin{aligned} &(1)\ \text{Solve } DF[i](x)v = e^n, \\ &(2)\ \sigma := d \cdot \operatorname{sgn}(\det DF[i](x)), \\ &(3)\ Tx := \sigma v / \| v \|_2, \end{aligned} \tag{1.6}$$

where $\| \cdot \|_2$ is the Euclidean norm, and $d = \pm 1$ is a given direction. Note that the matrix (1.5) is nonsingular for any index i, $1 \leq i \leq n$, for which the component $(Tx)_i$ is nonzero.

The process uses a local parameterization of the solution curve. Normally the continuation parameter is the variable x_i, for which the component $|(Tx)_i|$ is maximal. But in the case of certain curvature changes, where it appears that a limit point for this variable x_i is approaching, other choices of the continuation parameter are used.

If x denotes the current point, then prediction takes place along the Euler line

$$\pi(h) = x + hTx. \tag{1.7}$$

The choice of the step length h takes into account the quality of the corrector iteration during the computation of x, as well as a prediction of the change in curvature of the solution curve. Moreover, h is adjusted such that the (secant) distance between x and the next computed point will be approximately equal to h.

The corrector iteration starts from the predicted point $p = \pi(h)$ and solves the augmented equations

$$F[i]x = p_i e^n. \tag{1.8}$$

The user may specify as corrector iteration either a full Newton process or a modified Newton process with fixed Jacobian $DF[i](p)$ at the predicted point.

2. OUTLINE OF THE ALGORITHM

During the following description, we assume that we 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 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.

Step 1 For **KSTEP** > 0, the code goes to step 2. On the first call to **PITCON( )** for a given problem (**KSTEP** $= -1$ or **KSTEP** $= 0$) problem-dependent constants are set and user-control parameters are loaded or defaults used. If **KSTEP** $= 0$, the program then proceeds to step 2. If **KSTEP** $= -1$, the user requests that the input starting point **XR(*)** be checked for the condition $|\mathbf{F(XR)}| \le \frac{1}{2}$ **ABSERR**. If this is not the case, the corrector process is applied to the point **XR(*)** until the error condition is satisfied, or a failure has occurred. An unimprovable point results in a return of **IRET** $= -6$. If the starting point **XR(*)** was improved, the program returns with **IRET** $= 0$ and **KSTEP** $= 0$. If **KSTEP** $= 0$, the continuation loop begins with the starting point **XR(*)** stored in **XL(*)** and **XC(*)**, the step size **HTANCF** set to the input value of **H**, and the continuation parameter set to the input value of **IPC**. For **KSTEP** > 0, these quantities are computed and updated by the program itself.

Step 2 Target point check. If **IT** $\neq 0$, a target point is desired. The values of **XL(IT)** and **XC(IT)** are compared to **XIT**. If the target value is between these two values, the program computes the target point, sets **IRET** $=$ 1, and returns, temporarily interrupting normal continuation.

Step 3 Tangent and local continuation parameter calculation. If the loop was suspended at the last call to **PITCON( )** to allow the return of a limit point, then the tangent has already been calculated and a limit point check is superfluous, so the program skips to step 5. Otherwise, a vector in the tangent plane at **XC(*)** is computed. Suppose that the previous continuation parameter index was **IPL**, where on the first step **IPL** is user supplied. The new tangent is normalized, and the **IPL**-th component is forced to have the same sign as the **IPL**-th component of the previous tangent (or on first step, the same sign as the user input direction **DIRIPC**). Then the local continuation parameter **IPC** is determined. **IPC** is set to the location of the largest component of the tangent vector, unless a limit point for this choice appears to be approaching, in which case the location of the second largest component may be tried. Once **IPC** is set, certain quantities for step-length determination are computed.

Step 4 Limit point check. If **LIM**$\neq$0, the **LIM**-th components of the old and new tangents are compared. If these differ in sign, a limit point lies between **XL(*)** and **XC(*)**. The program attempts to find this limit point. If found, it stores the limit point in **XR(*)**, the tangent at **XR(*)** in **TL(*)**, sets **IRET**=2, and returns, temporarily interrupting the normal loop.

Step 5 Step-length computation. The program computes **HTANCF**, the step size to be used along the tangent to obtain the predicted point **XPRED(*)** $=$ **XC(*)** $+$ **HTANCF*TC(*)**, the starting point for the corrector process. In computing **HTANCF**, certain curvature and step-size data are updated.

Step 6 Prediction and correction step. With the predicted point **XPRED(*)** as a starting point, the corrector process is applied to correct the point **XCOR(*)** until both the residual ‖ **F(XCOR(*))** ‖ and the last corrector step ‖ **XSTEP(*)** ‖ are sufficiently small. 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 step size **HTANCF** is reduced and the corrector step is attempted again. If the step size shrinks below **HMIN**, the program sets an error flag and returns.

Step 7 Storing information before return. After a successful continuation step, the program rearranges its storage so that the entries corresponding to **XC(*)** and **XF(*)** hold the proper data, computes **CORDXF**, the size of the correction to the predicted point, and modifies **CORDXF** to a value that would correspond to an optimal number of corrector steps.

On normal return, the vector **XR(*)** contains a solution point on the curve (1.2), and is either a continuation point, a target point, or a limit point, which is indicated by the value of **IRET**. If **IRET** is negative, an error has occurred. If a limit point is returned, the tangent vector at the limit point is contained in the location **TL(*)**. On first call, the user must set some of the scalar parameters, and the starting point **XR(*)**. Thereafter, only **IT** and **XIT** should be changed by the user during a problem run.

If a new problem is to be run (whether a different function, or the same function with different starting point or error controls), the program may be reset by using **KSTEP** = −1 or 0, at which time the scalars and the point **XR(*)** must be set again. Note that in this case the statistical data in the common blocks /**COUNT1**/ and /**COUNT2**/ will be reset to 0 as well.

3. ORGANIZATIONAL DETAILS

There are five basic subroutines: **PITCON( )**, **CORECT( )**, **TANGNT( )**, **ROOT( )**, and **FSOLVE( )**. The user need only call **PITCON( )**. In addition, the code uses internally eight subroutines from the LINPAK package [1] and the BLAS package [4], namely, **ISAMAX( )**, **SAXPY( )**, **SCOPY( )**, **SDOT( )**, **SNRM2( )**, **SSCAL( )**, **SGEFA( )**, and **SGESL( )**. **PITCON( )** and **SNRM2( )** contain machine-dependent constants for which appropriate statements must be chosen.

The user must supply two subroutines of the form **FXNAME (NVAR, X, FX)** and **FPNAME (NVAR, X, FPRYM, NROW, NCOL)** with the actual names of these subroutines being passed as external quantities. Subroutine **FXNAME( )** evaluates the mapping F at the point **X(*)** in R^n, n = **NVAR**, and returns the results in **FX(*)**. Subroutine **FPNAME( )** evaluates the Jacobian matrix $DF(x)$ of dimension $(n - 1) \times n$, n = **NVAR**, at the point $x = X$ and returns it in the first $n - 1$ rows of the $n \times n$ array **FPRYM(*)**. If $DF(x)$ is not accessible, it is possible to supply in **FPNAME( )** some finite difference approximation of $DF(x)$. But the results will depend on the quality of this approximation and may be unreliable.

All calls of **FPNAME( )** and all solutions of the augmented equations occurring in (1.6) and (1.8) are handled by the subroutine **FSOLVE( )**. The subroutine included in the code uses full-matrix storage and hence limits the applications of the package to low-dimensional problems. It is easy to modify **FSOLVE( )** for the case of large, sparse problems by using instead of **SGEFA( )** and **SGESL( )** some appropriate decomposition and backsubstitution programs. For example, the Yale sparse matrix package [2] has been used for this purpose, but other codes can be applied as well. We refer also to [7] for an approach in the case in which the first $n - 1$ column of $DF(x)$ has a band-form.

The codes have been tested on a large number of problems. For some computational results, we refer to [8].

REFERENCES

1. DONGARRA, J.J BUNCH, J.R., MOLER, C B., AND STEWART, G.W *LINPACK User's Guide* Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1979.
2. EISENSTAT, S C., GURSKY, M.C., SCHULTZ, M.H., AND SHERMAN, A.H Yale sparse matrix package. Res. Reps 112 and 114, Yale University, Department of Computer Science, New Haven, 1977.
3. DEN HEIJER, C., AND RHEINBOLDT, W.C. On steplength algorithms for a class of continuation methods *SIAM J Numer Anal. 18* (1981), 925-947.
4. LAWSON, C.L., HANSON, R.J., KINCAID, D.R., AND KROGH, F.T Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Soft 5*, 3(Sept. 1979), 308-323.
5. MILNOR, J.W. *Topology from a Differential Viewpoint.* University of Virginia Press, Charlottesville, 1965
6 RHEINBOLDT, W.C Solution fields of nonlinear equations and continuation methods. *SIAM J. Numer. Anly 17* (1980), 221-237.
7. RHEINBOLDT, W.C. Numerical analysis of continuation methods for nonlinear structural problems. *Comput Struct 13* (1981), 103-114
8 RHEINBOLDT, W C , AND BURKARDT, J.V A locally parameterized continuation process *ACM Trans. Math. Softw 9*, 2(June 1983), 215-235

ALGORITHM

[A part of the listing is printed here. The complete listing is available from the ACM Algorithms Distribution Service (see page 269 for order form).]

```
      SUBROUTINE PITCON(NVAR, LIM, IT, XIT, KSTEP, IPC, IPCFIX, DIRIPC,
     * HTANCF, IRET, MODCON, IPIVOT, HMAX, HMIN, HFACT, ABSERR, RELERR,
     * RWORK, ISIZE, NROW, NCOL, FXNAME, FPNAME, SLNAME, LUNIT)
C
C***********************************************************************
C
C  1.  INTRODUCTION
C
C    THIS IS THE 30 JUNE 1982 VERSION OF PITCON,
C  THE UNIVERSITY OF PITTSBURGH CONTINUATION PACKAGE.
C  THIS VERSION USES SINGLE PRECISION AND FULL MATRIX STORAGE.
C
C    THIS PACKAGE WAS PREPARED WITH THE PARTIAL SUPPORT OF
C  THE NATIONAL SCIENCE FOUNDATION, UNDER GRANT MCS-78-05299,
C  BY WERNER C. RHEINBOLDT AND JOHN V. BURKARDT,
C  UNIVERSITY OF PITTSBURGH, PITTSBURGH, PA 15261.
C
C    SUBROUTINE PITCON COMPUTES POINTS ALONG A SOLUTION CURVE OF AN
C  UNDERDETERMINED SYSTEM OF NONLINEAR EQUATIONS OF THE FORM FX=0.
C  THE CURVE IS SPECIFIED TO BEGIN AT A GIVEN STARTING SOLUTION
C  X OF THE SYSTEM.  HERE X DENOTES A REAL VECTOR OF NVAR
C  COMPONENTS AND FX A REAL VECTOR OF NVAR-1 COMPONENTS.
C  NORMALLY EACH CALL TO PITCON PRODUCES A NEW POINT FURTHER ALONG
C  THE SOLUTION CURVE IN A USER-SPECIFIED DIRECTION.
C
```

```
C    AN OPTION ALLOWS THE SEARCH FOR AND COMPUTATION OF TARGET POINTS,  PIT  260
C  THAT IS, SOLUTION POINTS X FOR WHICH X(IT) = XIT FOR SOME USER       PIT  270
C  SPECIFIED VALUES OF IT AND XIT.                                      PIT  280
C                                                                       PIT  290
C    A FURTHER OPTION ALLOWS THE SEARCH FOR AND COMPUTATION OF LIMIT    PIT  300
C  POINTS FOR SPECIFIED COORDINATE LIM, THAT IS, SOLUTION POINTS FOR    PIT  310
C  WHICH THE LIM-TH COMPONENT OF THE TANGENT VECTOR IS ZERO.            PIT  320
C                                                                       PIT  330
C    EXPLANATIONS OF THE ALGORITHMS USED IN THIS PACKAGE MAY            PIT  340
C  BE FOUND IN                                                          PIT  350
C                                                                       PIT  360
C  WERNER RHEINBOLDT,                                                   PIT  370
C  SOLUTION FIELD OF NONLINEAR EQUATIONS AND CONTINUATION METHODS       PIT  380
C  SIAM JOURNAL OF NUMERICAL ANALYSIS, 17, 1980, PP 221-237             PIT  390
C                                                                       PIT  400
C  COR DEN HEIJER AND WERNER RHEINBOLDT,                                PIT  410
C  ON STEPLENGTH ALGORITHMS FOR A CLASS OF CONTINUATION METHODS,        PIT  420
C  SIAM JOURNAL OF NUMERICAL ANALYSIS 18, 1981, PP 925-947              PIT  430
C                                                                       PIT  440
C  WERNER RHEINBOLDT,                                                   PIT  450
C  NUMERICAL ANALYSIS OF CONTINUATION METHODS FOR NONLINEAR             PIT  460
C  STRUCTURAL PROBLEMS,                                                 PIT  470
C  COMPUTERS AND STRUCTURES, 13, 1981, PP 103-114                       PIT  480
C                                                                       PIT  490
C***********************************************************************PIT  500
```