## Abstract

We present a new method for computing *simple turning points* of nonlinear equations of the form $G(u,\lambda) = 0$ which is based on applying Newton's method to the characterization $d\lambda(\sigma)/d\sigma = 0$, where $\sigma$ is a *pseudo-arclength* parameter used in a continuation method for following the solution paths. The method is *quadratically* convergent and needs only *one* starting point on the solution path. Second derivatives of G (or difference approximations of them) have to be computed but the method is relatively insensitive to their values and they also give rise to a more accurate *second* order predictor in the continuation method. We present a *chord*-Newton variant for improving the efficiency of the algorithm which requires only *one* factorization of a Jacobian matrix. We also present a *damped*-Newton variant for improving the *robustness* and the *global* convergence of the algorithm. Results of extensive numerical experiments on two standard nonlinear elliptic problems of Simpson's [24] show that the new algorithm compares favourably with the best of the existing methods in terms of efficiency and robustness.


# Newton-Like Pseudo-Arclength Methods for Computing Simple Turning Points


**Tony F. Chan**[1]



April, 1982

Technical Report #233


---


[1]Computer Science Department, Yale University, Box 2158, Yale Station, New Haven, CT 06520.


The author's work was supported by Department of Energy Contract DE-ACO2-81ER10996.

## Table of Contents

## 1. Introduction

Many problems in computational physics can be formulated as *nonlinear eigenvalue problems* of the form:

$$G(u,\lambda) = 0, \tag{1}$$

where $u \in B$ (a real Banach space), $\lambda \in R$, and G is a continuously differentiable operator mapping B x R into B. Usually, u represents the *solution* to the physical problem (e.g. flow field, structural displacement) and $\lambda$ is related to a physical parameter (e.g. Reynold's number, load on a structure). Often, one is interested in the dependence of the solution $u(\lambda)$ on the parameter $\lambda$, i.e. in tracing the *solution branches* $[u(\lambda), \lambda]$ of (1). When the operator G is *nonlinear* in u and $\lambda$, this is usually accomplished numerically by some version of Newton's method applied to (1) for a fixed value of $\lambda$, which makes use of the Jacobian matrix $G_u(u,\lambda)$. However, the solution branches often possess very interesting but complicated nonlinear bifurcation behaviours, among which are existence of *multiple* solutions and *singular* points (where $G_u(u,\lambda)$ is singular) known as *turning points* (where the solution branch bends back on itself) and *bifurcation points* (where two or more solution branches cross.) Straightforward application of Newton's method to (1) encounters difficulties near these singular points. To overcome these difficulties, some kind of path following continuation method [2, 10, 14, 18] is usually employed. These continuation methods are designed to trace past turning points and can be modified to switch branches at bifurcation points.

In many applications, in addition to tracing the solution branches, one is also interested in locating the singular points themselves, because they are often related to the *stability* of the solution. Due to their special physical significance, many algorithms have been proposed for determining these singular points accurately. In this paper, we shall only deal with the determination of *simple turning points*, which can be characterized as points on the solution curve where

$$G_u \text{ is singular, and} \tag{2}$$

$$G_\lambda \notin \text{Range}(G_u). \tag{3}$$

For an excellent survey of existing methods for computing turning points, we refer the reader to the report by Melhem and Rheinboldt [13] in which they compared the performances of algorithms proposed by Abbott [1], Moore and Spence [15], Seydel [23], Paumier [16], Pönisch and Schwetlick [17], Rheinboldt [19, 20], Schwetlick [22] and Simpson [24]. These methods can be classified into two general classes. The first consists of local iterative algorithms based on an inflated

system consisting of (1) augmented by a characterization similar to (2), constructed so that the turning point is a *unique and isolated* solution of the inflated system. The other class of algorithms consists of methods based on a path tracing continuation method by successively using it to compute points on the solution curve that approach the turning point. These algorithms can further be categorized by whether one or more points on the solution curve are needed as initial guess and whether second derivatives of G are needed. In Table 1-1, we tabulated the properties of the best methods as found by Melhem and Rheinboldt, judged by an overall measure of excellency in terms of a combination of efficiency, robustness and generality. For the purpose of comparison, we have included the method that we are proposing in this paper.

**Table 1-1:** Properties of Some Methods for Computing Turning Points

| | Initial Points | Rate of Convergence | Needs 2nd Derivative? | Class of Method | Characterization of Turning Point |
|---|---|---|---|---|---|
| Abbott | 1 | 2 | yes/no | I | $\dot{\lambda}(s) = 0$ |
| Moore & Spence | 1 | 2 | yes/no | I | $G_u w = 0,\ w \neq 0$ |
| Pönisch & Schwetlick | 1 | 2 | yes/no | I | $\dot{\lambda}(s) = 0$ |
| Rheinboldt | 2 | 1.618 | no | C | $\dot{\lambda}(s) = 0$ |
| Schwetlick | 2 | 1.618 | no | C | $\lambda(\tau)$ = extremum |
| Chan | 1 | 2 | yes/no | C | $\lambda'(\sigma) = 0$ |

Notation: I : Inflated System, C : Continuation,
s is the arclength parameter, $\tau$ and $\sigma$ are arclength-like parameters,
"yes/no" means that the basic method requires second derivatives
of G but difference approximations are also used by the authors.

As can be seen from the table, the method that we are proposing is *quadratically* convergent, needs only *one* initial guess on the solution curve, and is based on an underlying continuation method for branch tracing. Methods based on continuation have certain desirable properties. First, they can

build upon the curve tracing capabilities that are already in the continuation procedure. For example, very often the *same* linear equation solver can be used without having to refactor any Jacobian matrix. Second, they naturally provide more details of the solution curve around the turning points. Lastly, as we shall demonstrate later, requiring the iterates to lie on the solution curve tends to make the algorithms more robust than methods based on using augmented systems. The property of requiring only *one* initial point is also desirable because most continuation methods have to be slowed down near turning points and it may be relatively expensive to obtain two points on the solution curve where $\dot{\lambda}$ changes sign, as is needed by some methods. We shall review briefly the formulation of typical continuation methods in Section 2.

Many of the methods in Table 1-1 uses the characterization $\dot{\lambda}(s) = 0$ for turning points. Our method is based on an alternative characterization of simple turning points, namely, that

$$\lambda'(\sigma) \equiv d\lambda(\sigma)/d\sigma = 0 \tag{4}$$

where $\sigma$ is the *pseudo-arclength parameter* used in the continuation method. This characterization has been suggested by Keller [11] although he only considered *secant* methods and no numerical results were given. Our method is based on applying *Newton's* method to (4). We show in Section 3 that the second derivatives $[u''(\sigma), \lambda''(\sigma)]$ can be computed rather inexpensively if the second derivatives of G are available. We note that *none* of the methods based on continuation cited in [13] is quadratically convergent. This is obviously because the authors tried to avoid computing second derivatives, which in some applications are very difficult to obtain. However, since a function evaluation in these methods involves an *inner* Newton iteration which could be costly, we believe that in many applications where the second derivatives (or approximations of them) are available, the use of a method with a faster convergence rate may be benefitial. As we shall show in Section 4.1, the availability of second derivatives also leads to a much more accurate *predictor* in the underlying continuation method, which in turn improves the efficiency of the overall algorithm. Another desirable property for an algorithm is that of requiring only a solver for $G_u$ (rather than a matrix derived from $G_u$) since such a solver may already be available in the application discipline and it can also exploit special solution techniques (e.g. fast elliptic solvers). We show in Section 4.2 how this can be arranged in our algorithm. In Section 5, we present a *chord*-Newton variant for improving the *efficiency* and a *damped*-Newton variant for improving the *robustness* and *global* convergence of the basic algorithm. In Section 6, we discuss briefly the work and storage

requirements of the new algorithm, which are comparable to most of the existing methods. Extensive numerical experiments have been performed on applying the algorithm and its variants to two standard nonlinear elliptic problems of Simpson's [24] and the results are presented in Section 7. They demonstrate that the new algorithm is both efficient and robust and compares favourably with the best of the existing methods on these two problems.

## 2. Pseudo-Arclength Continuation

In this section, we review the essential features of some common path-following continuation methods.

The key idea is to *parametrize* the solutions $[u(\sigma),\lambda(\sigma)]$ in terms of a new parameter $\sigma$ that approximates the *arclength* parameter s, instead of parametrizing $u(\lambda)$ in terms of the natural parameter $\lambda$. This is usually achieved by *augmenting* the equation (1) by an auxillary equation that approximates the *arclength* condition:

$$||\dot{u}(s)||^2 + |\dot{\lambda}(s)|^2 = 1 \ , \tag{5}$$

to give an *inflated* system with unknowns $u(\sigma)$ *and* $\lambda(\sigma)$:

$$\begin{aligned} G(u(\sigma),\lambda(\sigma)) &= 0 \ , \\ N(u(\sigma),\lambda(\sigma),\sigma) &= 0 \ . \end{aligned} \tag{6}$$

Instead of solving for $u(\lambda)$ for a given value of $\lambda$, we solve for $u(\sigma)$ and $\lambda(\sigma)$ for a given value of $\sigma$. Newton's method and its variants are usually used to solve (6), in which case we need to solve *linear* systems with the following inflated matrix:

$$M = \begin{vmatrix} + G_u & G_\lambda + \\ + N_u & N_\lambda + \end{vmatrix} \ . \tag{7}$$

The auxillary function N is constructed so that the matrix M is *nonsingular* on the solution branch, *even near or at turning points*. Thus, Newton's method encounters no difficulties with this inflated system and quadratic convergence is achievable.

Another major component of a continuation method is the computation of the *unit tangent* $[\dot{u}(s),\dot{\lambda}(s)]$ to the solution curve at a point $[u,\lambda]$ on the solution curve, which can be computed

relatively inexpensively from its definition:

$$G_u(u,\lambda)\dot{u} + G_\lambda(u,\lambda)\dot{\lambda} = 0$$
$$||\dot{u}||^2 + |\dot{\lambda}|^2 = 1\ , \tag{8}$$

by solving only *one* linear system with $G_u$. The system (8) determines $[\dot{u},\dot{\lambda}]$ up to an directional orientation, which can be fixed by some convention. The tangent is usually used in a first order *predictor* to obtain an initial guess for the Newton iteration applied to the system (6).

We summarize the essential features in the following general algorithm:

---

**Algorithm PAC**$[u_0,\lambda_0,\sigma,u(\sigma),\lambda(\sigma)]$

**Pseudo-Arclength Continuation** : Given $[u_0,\lambda_0]$ on the solution curve, and a step length $\sigma$, computes the new solution $[u(\sigma),\lambda(\sigma)]$ satisfying (6).

1. Compute the unit tangent $[\dot{u}_0,\dot{\lambda}_0]$ at $[u_0,\lambda_0]$ by (8).
2. Compute the *predicted* solution $[u_p,\lambda_p]$ given by :
$$u_p = u_0 + \sigma\dot{u}_0,$$
$$\lambda_p = \lambda_0 + \sigma\dot{\lambda}_0. \tag{9}$$
3. Use $[u_p,\lambda_p]$ as initial guess in a Newton-like iteration for solving the system (6) to obtain $[u(\sigma),\lambda(\sigma)]$.

---

A few typical N's that have been used in the literature are:

1. $N_1(u,\lambda,\sigma) \equiv \dot{u}_0^T(u - u_0) + \dot{\lambda}_0(\lambda - \lambda_0) - \sigma$ (introduced by Keller [10]),
2. $N_2(u,\lambda,\sigma) \equiv e_i^T(y - y_0) - \sigma$, where $y = (u,\lambda)^T$, $e_i$ is the i-th unit vector and the index i is chosen so that the matrix M is as well-conditioned as possible (introduced by Abbott [1], Kubicek [12] and Rheinboldt [18]).

## 3. Newton on $\lambda'(\sigma) = 0$

We shall consider only *simple* turning points where the nullity of $G_u$ is one. Consider the situation where we have an approximation $[u_0,\lambda_0]$ to a turning point $[u_*,\lambda_*]$. The method that we are proposing works by *estimating the step length* $\sigma$ to use in applying one step of the pseudo-arclength continuation procedure PAC$[u_0,\lambda_0,\sigma,u(\sigma),\lambda(\sigma)]$ so that $u(\sigma) \equiv u_*$ and $\lambda(\sigma) \equiv \lambda_*$. The basis for estimating $\sigma$ is derived from the following characterization of simple turning points:

**Definition 1:** Define $\lambda'(\sigma) \equiv d\lambda(\sigma)/d\sigma$ and $u'(\sigma) \equiv du(\sigma)/d\sigma$.

**Theorem 2:** Assume $N_\sigma(\sigma) \neq 0$. Then $\lambda'(\sigma) = 0$ *if and only if* $[u(\sigma),\lambda(\sigma)] \equiv [u_*,\lambda_*]$.

**Proof:** First note, by differentiating (6) by $\sigma$, that $[u'(\sigma),\lambda'(\sigma)]$ satisfies :

$$\begin{pmatrix} G_u(\sigma) & G_\lambda(\sigma) \\ N_u(\sigma) & N_\lambda(\sigma) \end{pmatrix} \begin{pmatrix} u'(\sigma) \\ \lambda'(\sigma) \end{pmatrix} = \begin{pmatrix} 0 \\ -N_\sigma(\sigma) \end{pmatrix}, \tag{10}$$

where the coefficient matrix in (10) is *nonsingular* by construction. Thus $[u'(\sigma),\lambda'(\sigma)]$ is *well-defined* even near or at a turning point. Now first assume $\lambda'(\sigma) = 0$. The second equation in (10) implies that $N_u u' = -N_\sigma \neq 0$. Thus $u'$ is nontrivial. The first equation in (10) reduces to $G_u u' = 0$. Since $u'$ is nontrivial, $G_u$ must be singular. Next assume that $[u(\sigma),\lambda(\sigma)] \equiv [u_*,\lambda_*]$. Then by (2) $G_u(\sigma)$ is singular. If $\lambda'(\sigma) \neq 0$, then the first equation in (10) implies that $G_\lambda(\sigma) \in \text{Range}(G_u(\sigma))$, which contradicts (3).

□

We note that both $N_1$ and $N_2$ satisfy the hypothesis of Theorem 2.

Theorem 2 provides a basis for estimating the step length $\sigma$ because it reduces the problem to one of finding a *root* of $\lambda'(\sigma) = 0$. Our method is based on applying Newton's method for doing this, for which we need to compute $\lambda''(\sigma)$. It is not surprising that this requires computing the second derivatives of G. By differentiating (10) with respect to $\sigma$, we obtain the following system for computing $[u''(\sigma),\lambda''(\sigma)]$:

$$\begin{pmatrix} G_u(\sigma) & G_\lambda(\sigma) \\ N_u(\sigma) & N_\lambda(\sigma) \end{pmatrix} \begin{pmatrix} u''(\sigma) \\ \lambda''(\sigma) \end{pmatrix} = \begin{pmatrix} -(G_{uu}(\sigma)u'(\sigma)u'(\sigma) + 2\,G_{u\lambda}(\sigma)u'(\sigma)\lambda'(\sigma) + G_{\lambda\lambda}(\sigma)\lambda'(\sigma)\lambda'(\sigma)) \\ -(N_{uu}(\sigma)u'(\sigma)u'(\sigma) + 2\,N_{u\lambda}(\sigma)u'(\sigma)\lambda'(\sigma) + N_{\lambda\lambda}(\sigma)\lambda'(\sigma)\lambda'(\sigma)) \end{pmatrix}, \tag{11}$$

$$- N'_{\sigma\sigma}(\sigma)$$

We note that the systems governing $[u',\lambda']$ and $[u'',\lambda'']$ have identical coefficient matrices, which are almost exactly the *same* as that used in the last step of the Newton iteration in the pseudo-arclength procedure. Thus, the same factorization of these matrices can be used to compute $[u',\lambda']$ and $[u'',\lambda'']$ and they can be obtained essentially *free*.

We outline the basic version of our method in algorithmic form:

---

**Algorithm NTP$[u_0,\lambda_0,u_*,\lambda_*]$**

**Newton's Method for Locating Turning Points:** Starting with an initial guess $[u_0,\lambda_0]$ on the solution curve, computes an approximation $[u_*,\lambda_*]$ to a turning point.

Initialize $\sigma = 0$.

Loop until convergence:

1. Compute $[u'(\sigma),\lambda'(\sigma)]$ and $[u''(\sigma),\lambda''(\sigma)]$ by (10) and (11).
2. Compute the *change* in the step length $\delta\sigma = -\lambda'(\sigma)/\lambda''(\sigma)$.
3. Update the new step length $\sigma \Leftarrow \sigma + \delta\sigma$.
4. Call PAC$[u_0,\lambda_0,\sigma,u(\sigma),\lambda(\sigma)]$.
5. Set $u_* \Leftarrow u(\sigma)$, $\lambda_* \Leftarrow \lambda(\sigma)$.

---

Under mild conditions on the smoothness of G, it is not difficult to prove that if $[u_0,\lambda_0]$ is close enough to a turning point $[u_*,\lambda_*]$, Algorithm NTP produces iterates $[u(\sigma),\lambda(\sigma)]$ that converge *quadratically* to $[u_*,\lambda_*]$. We shall not pursue this analysis here.

We note that $[u',\lambda']$ is *not* equal to the unit tangent $[\dot{u},\dot{\lambda}]$ but is a *scaled* version of it. As can be seen from Table 1-1, the characterization $\dot{\lambda}(s) = 0$ for turning points have been used by many authors but Keller [11] seems to be the only one who has considered the use of the *pseudo-arclength parameter* $\sigma$ of a continuation procedure, together with the characterization $\lambda'(\sigma) = 0$, in the context of an algorithm for finding turning points. A system similar to (11) has been derived by Pönisch and Schwetlick [17] but their method is not based on a continuation procedure.

## 4. Implementation

In this section, we discuss some of the implementation details for Algorithm NTP. We address three issues: the construction of a more accurate predictor, algorithms for solving linear systems with the inflated matrix of the form (7), and the use of difference approximations for second derivatives.

### 4.1. Second Order Predictor

Instead of the *first* order predictor used in Step (2) of Algorithm PAC, which is called by Step (4) of Algorithm NTP, the availability of second derivatives allow the use of the following more accurate predictor:

$$u_p = u(\sigma) + (\delta\sigma)u'(\sigma) + (\delta\sigma)^2 u''(\sigma)/2 \ ,$$
$$\lambda_p = \lambda(\sigma) + (\delta\sigma)\lambda'(\sigma) + (\delta\sigma)^2 \lambda''(\sigma)/2 \ . \qquad (12)$$

This new predictor is more accurate for two reasons. First, the current approximation $[u(\sigma),\lambda(\sigma)]$ to the turning point is used instead of $[u_0,\lambda_0]$, and second, it has *second* order accuracy. Note that this more accurate predictor is essentially *free*, since all the quantities in (12) have already been computed in Algorithm NTP before the call to Algorithm PAC. Its higher accuracy greatly reduces the cost of the inner Newton iteration in Step (4) of Algorithm PAC.

### 4.2. Block Elimination and Deflation Techniques

All the linear systems that arise in our algorithm are of the form:

$$M \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} A & b \\ c^T & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \qquad (13)$$

where the n by n matrix A may become singular near a turning point but the vectors b and c are chosen so that M remains nonsingular and well-conditioned. The algorithm that we have chosen to use for solving the linear systems of the form (13) is the following *block-elimination* algorithm :

---

**Algorithm BE:** [5, 10]

(1) Solve $A\,v = b,$ (14)

$A\,w = f.$ (15)

(2) Compute $y = (g - c^T w) / (d - c^T v).$ (16)

(3) Compute $x = w - y\,v.$ (17)

---

The work consists mainly of one factorization of A and two backsolves with the LU factors of A. If there are many right hand sides with the same matrix M, then the factorization of A and the vector v need only be computed once, and the work reduces to only one backsolve for each right hand side, which makes Algorithm BE extremely attractive in such cases. These situations arise in the chord-Newton variant of Algorithm NTP (see Section 5.1). Note also that only a solver for A is needed, and therefore any special structures (e.g. sparsity, bandedness, special data structures) in A can be exploited and special solvers for A can be used (e.g. fast direct solvers, multi-grid solvers). However, as we have shown in [3], Algorithm BE maybe ***unstable*** numerically when A is nearly singular, as is the case in the present application. The main source of instability is in Step (1) of Algorithm BE where the vectors v and w are computed inaccurately when A is nearly singular. In [3], we proposed using *implicit deflation* techniques developed in [4, 25] to compute accurate representations for the solutions v and w. These deflation techniques can be viewed as working in subspaces orthogonal to approximate null vectors of A and are *implicit* in the sense that they only involve solving systems with A. We then use these *deflated decompositions* of v and w to obtain a stable variant of the BE algorithm, which we called **Algorithm DBE**. The only overhead involved for performing the deflation in this algorithm is the computation of two approximate left and right *null vectors* for A. These can be obtained either by an inverse power method or by a technique based on the existence of a small pivot in the LU-factorization of A [6]. In any case, the extra work amounts to only a few backsolves, which is usually negligible in comparison with the work involved in computing the factorization. We refer the details of Algorithm DBE to [3], where we also presented a *backward error analysis* that shows that it is numerically stable.

We have assumed that *direct* elimination methods are used for solving the linear systems that

arise. For the use of *iterative methods*, which might be more attractive for large and sparse problems, we refer the reader to [7]. For another method for solving the inflated systems, see [21].

### 4.3. Difference Approximations for 2nd Derivatives

In the context of algorithms for computing turning points, for *any* method to achieve *quadratic* convergence, second derivatives of G are required in general. Unfortunately, in many applications, second derivatives are difficult to compute or not available at all. For this reason, many algorithms avoid using second derivatives explicitly. In the specific context of using the characterization $\lambda'(\sigma) = 0$ for locating turning points, there are at least three ways to achieve this. The first is to use a *secant-like* method for finding a zero of $\lambda'(\sigma)$, as is the case in the methods of Keller [11] and Rheinboldt [19, 20]. However, the convergence rate will then not be quadratic. In order to retain quadratic convergence, at least *approximately*, we choose to work with a Newton-like method similar to Algorithm NTP. Within this context, there are at least two ways to avoid second derivatives. The first is to use a difference approximation for $\lambda''(\sigma)$, by evaluating $\lambda'(\sigma)$ at two adjacent points. This is essentially the approach taken by the method of Abbott in Table 1-1. Note that each evaluation of $\lambda'(\sigma)$ may be rather costly as it involves solving a few systems with the inflated matrix M. The last approach, which is the one we have adopted in this paper, is to use a difference approximation for computing the second derivatives of G. Note that these appear only on the *right hand side* of (11) rather than in the coefficient matrix, as is the case in the method of Moore and Spence [15]. We believe that this property of the algorithm leads to better numerical stability. For the numerical experiments in this paper, we have used a simple centered difference approximation. For example, $G_{\lambda\lambda}(\sigma)$ is approximated by:

$$G_{\lambda\lambda}(\sigma) \approx (G_{\lambda}(u(\sigma),\lambda(\sigma)+\epsilon) - G_{\lambda}(u(\sigma),\lambda(\sigma)-\epsilon)) \,/\, 2\epsilon \,. \tag{18}$$

In practice, one can use better techniques; see for example [9].

## 5. Variants

In this section, we present variants of the basic Algorithm NTP designed to improve its efficiency and robustness.

### 5.1. Chord-Newton Variant

With *direct* methods for solving the linear systems, the most expensive part of the computation is usually in *factoring* the Jacobian matrix $G_u$. Therefore, one can save a great deal of computation by *reusing* the factors of a nearby matrix. There are *two* Newton iterations involved in Algorithm NTP, both of which allow *chord*-Newton variants. For the *outer* Newton iteration, it does *not* pay to use the chord-Newton variant because the coefficient matrix governing $[u''(\sigma),\lambda''(\sigma)]$ in (11) is the *same* as the one governing $[u'(\sigma),\lambda'(\sigma)]$ in (10). Thus, the second derivatives $[u''(\sigma),\lambda''(\sigma)]$ can be computed very inexpensively by performing only *one* back-substitution. For the *inner* iteration in Algorithm PAC, however, one can obtain a chord-Newton variant by using the *same* LU-factors of a Jacobian matrix $G_u$ in *all* the Newton steps, for example, by using the LU-factors of the matrix $G_u$ used in computing $\lambda'(\sigma)$ and $\lambda''(\sigma)$ in the Step (1) of Algorithm NTP.

We note that with Algorithm DBE, $G_\lambda$ can be updated in $M(\sigma)$ in *each* step of the chord-Newton iteration *without* incurring a factorization of $G_u$, which in general gives a better approximation for $M(\sigma)$ than if an old copy of $G_\lambda$, say $G_\lambda(0)$, were used. However, if we choose *not* to update $G_\lambda$, then the vector v in Algorithm DBE can be computed *once for all* and each solve with M then involves only *one*, rather than *two*, solve with $G_u$. Therefore, if $G_\lambda$ does not change very much around the turning point, it might be more efficient *not* to update $G_\lambda$ at every step in the chord-Newton iteration. This is the strategy that we have used in our numerical experiments.

The above chord-Newton variant requires *one* factorization of $G_u$ *per outer iteration step*. This is similar to the treatment of the chord version of Rheinboldt's method. However, if the initial guess $[u_0,\lambda_0]$ is close enough to the turning point, one can reduce the work further. We can factor $G_u$ *once only* at the initial guess $[u_0,\lambda_0]$ and reuse these factors of $G_u(u_0,\lambda_0)$ in all subsequent iteration steps, *both outer and inner*. However, for the convergence of the outer iteration, we have to ensure that the function values in the *outer* iteration, i.e. $\lambda'(\sigma)$, are evaluated accurately. Since the system (10) governing $[u'(\sigma),\lambda'(\sigma)]$ is *linear*, we can use the following *iterative improvement* algorithm for doing this:

Starting with an initial guess for $t(\sigma)$, iterate until convergence:

$$t(\sigma) \Leftarrow t(\sigma) + M(0)^{-1} (r(\sigma) - M(\sigma)\, t(\sigma)), \tag{19}$$

where

$$M(0) = \begin{pmatrix} G_u(0) & G_\lambda(0) \\ N_u(\sigma) & N_\lambda(\sigma) \end{pmatrix},$$

$$M(\sigma) = \begin{pmatrix} G_u(\sigma) & G_\lambda(\sigma) \\ N_u(\sigma) & N_\lambda(\sigma) \end{pmatrix},$$

$$r(\sigma) = ( 0 , -N_\sigma(\sigma))^T ,$$

and

$$t(\sigma) = ( u'(\sigma) , \lambda'(\sigma) )^T.$$

A similar algorithm can be applied to the $[u''(\sigma),\lambda''(\sigma)]$ system (11) as well. Moreover, since the second derivatives $[u''(\sigma),\lambda''(\sigma)]$ are available from the last outer Newton iteration, one can use a *first order predictor* for $[u'(\sigma),\lambda'(\sigma)]$ in (19), similar to the one used in (12). Furthermore, although we haven't pursued it here, the iteration (19) can also be *accelerated*, for example, by a conjugate gradient type method. No predictor for $[u'',\lambda'']$ is available, however, unless one stores previous values and uses extrapolation.

## 5.2. Damped-Newton Variant

It is well-known that Newton's method is only *locally* convergent. In the context of Algorithm NTP, if the initial guess $[u_0,\lambda_0]$ is far away from a turning point, then the step $\delta\sigma$ generated at Step (2) of Algorithm NTP may be so large that either there is no solution for $[u(\sigma),\lambda(\sigma)]$ or the inner Newton iteration in Algorithm PAC fails to converge. To improve the robustness of the algorithm, we consider the use of a *damped*-Newton variant. Since our algorithm is based on a continuation method, this can be arranged naturally by replacing Step (3) and (4) of Algorithm NTP by :

3'. **If** the previous $\delta\sigma$, say $\delta\sigma_p$, was *damped* **and** $|\delta\sigma| > |\delta\sigma_p|$ **then**

$\delta\sigma \Leftarrow \text{sign}(\delta\sigma)\, |\delta\sigma_p|$

4'. **Repeat until** *convergence*:

4.1 $\sigma \Leftarrow \sigma + \delta\sigma$.

4.2 **Call** PAC$[u_0,\lambda_0,\sigma,u(\sigma),\lambda(\sigma)]$.

4.3 **If** *no convergence* in PAC, **then** $\delta\sigma \Leftarrow \delta\sigma \,/\, \gamma$.

Here $\gamma$ is a scalar damping factor (we used $\gamma = 2$). To reduce the work wasted in the *damped* steps, we declare that Algorithm PAC has ***failed*** if either the number of iterations exceeds a maximum (we used a value of 5) or if the norm of the residuals $||G||$ is not *less* than that at the previous iteration. This is similar to the treatment in [18, 19]. Since the methods are based on a continuation procedure, it can be shown [18] that the loop in Step (4') above will terminate with a *nonzero* step length $\delta\sigma$. For methods based on inflated systems, no natural damped-Newton variant exists.

## 6. Work and Storage

In Table 6-1, we summarize the work and storage requirements of Algorithms NTP and and its chord variant, assuming that a direct factorization-solve method is used for solving the linear systems. The storage for the *damped*-Newton version is the same as that for the non-damped version. Its work is more difficult to estimate since it depends on exactly how the damped steps are taken. We have therefore not included it in the table.

**Table 6-1:** Work and Storage Per Step

| | Work / Step | | | | | Storage | |
|---|---|---|---|---|---|---|---|
| | | | EVALUATIONS | | | LU | |
| Algor. | Factors | Solves | Function | Jacobian | 2nd deriv | Factors | Vectors |
| True Newton | N + 1 | N + 2 | N + 1 | N + 1 | 1 | 1 | 9 |
| Chord Newton | 0 (1)* | N + I | N + 1 | N + 1 | 1 | 1 | 10 |

Notation:
N : number of iterations in Algorithm PAC.
I : number of iterative improvement iterations for computing $[u', \lambda']$ and $[u'', \lambda'']$.
*For the chord version, no factorization is needed after the first iteration.

Storage are needed for the vectors: u, $\delta u$ (in the inner Newton iteration), $G_\lambda$, $N_u$, G, u', u'' and the two approximate null vectors. For the chord version, one more vector is needed to store the old $G_\lambda$ or v. We have ignored the work involved in computing the approximate null vectors needed for deflation in Algorithm DBE since they have to be computed only *once* per factorization and the work is thus negligible in comparison to the factorization cost for $G_u$.

We note that the storage is comparable to those of methods of similar type surveyed in [13], except that a few more vectors are required. The work is also similar, except that for the chord version, no other author seems to have used the potentially more efficient iterative improvement algorithm (19) for computing $[u',\lambda']$ and $[u'',\lambda'']$ with only *one* factorization of $G_u$.

For a general, *dense* n by n problem, the work required for evaluating the second derivatives of G in our algorithm is $O(n^3)$. However, for many problems with sparsity (e.g. see Section 7), the work is usually much *less*.

## 7. Numerical Experiments

We have performed extensive experiments on applying our algorithm and its variants to the following nonliear elliptic eigenvalue problem [13, 15, 24]:

$$G(u,\lambda) \equiv \Delta u + F(u,\lambda) = 0, \tag{20}$$

on the unit square with zero Dirichlet boundary conditions. Following previous authors, we use a *fourth*-order finite difference discretization of (20) on a uniform mesh of size $h = 1/m$, which results in a system of $n \equiv (m-1)^2$ nonlinear equations. Two choices for the function F have been considered:

$$F_1 = \lambda\, e^u, \tag{21}$$

$$F_2 = \lambda\, (\, 1 + (u+u^2/2)/(1+u^2/100)\,). \tag{22}$$

For $m = 8$, the turning points that we are interested in are given in Table 7-1.

**Table 7-1:** Turning Points for $m = 8$

| F | $\lambda$ | u(0.5,0.5) |
|---|---|---|
| $F_1$ | 6.807504 | 1.391598 |
| $F_2$ | 7.980356 | 2.272364 |

All computations have been performed on a DEC-20, with 27-bit mantissas, corresponding to a relative machine precision of about .4 x $10^{-8}$. The matrices corresponding to $G_u$ are banded and are factored and solved by the LINPACK routines SGBCO and SGBSL [8]. The work for the factorization is $O(m^4)$, for the solve is $O(m^3)$ and for the evaluation of second derivatives is $O(m^2)$. Thus, for problems of this kind (generally differential equations with a local stencil), the cost of evaluating second derivatives is smaller than the cost of the solve phase.

We use the pseudo-arclength function $N_1$ in all our computations. We note that $N_1$ is *linear* in all its arguments and hence all its second derivatives vanish and $N_u(\sigma)$ and $N_\lambda(\sigma)$ are constants.

For the convergence of the Newton iteration in Algorithm PAC, we use the criteria: $||G|| < 10^{-5}$ *and* $||N|| < 10^{-5}$, which is adequate for the scale of our problems. For the iterative improvement algorithm (19), we stop if the *relative change* in the iterate is less than $10^{-5}$. For the difference approximations of second derivatives, we use a value of $\epsilon = 10^{-4}$ in (18). For computing the approximate null vectors needed in Algorithm DBE, we always use 3 steps of inverse iteration, the details of which can be found in [3]. The damped version is always used. We shall use the switch /FD to denote the use of difference approximations of second derivatives.

Following Melhem and Rheinboldt [13], we considered two starting points for $F_2$: $\lambda_0 = 7.96754$ and $\lambda_1 = 7.94617$ . We also considered two other starting points: $\lambda_3 = 7.5$ and $\lambda_4 = 7.0$ . All are on the *lower* branch of the solution curve.

We first tested the **Newton** version. In Tables 7-2, 7-3 and 7-4, we tabulate the results of applying Algorithm NTP to $F_2$, starting from $\lambda_0$, $\lambda_1$ and $\lambda_2$ respectively. In Table 7-5, we tabulate the results for $F_1$ starting from $\lambda = 6.8$ on the lower branch.

The **notation** is as follows:

- I : Number of outer Newton iteration.
- I1 : Number of iterative improvement iterations in computing $\lambda'$, chord version only.
- I2 : Number of iterative improvement iterations in computing $\lambda''$, chord version only.
- D : Number of damped-Newton steps.
- N : Number of inner Newton iteration in Algorithm PAC.

For comparison, we have included in the tables the values of $\dot{\lambda}(s)$, which are not needed in the algorithm.

**Table 7-2:** Results for $F_2$, initial guess $\lambda_0 = 7.96754$, true Newton

| I | $\lambda'$ | I1 | $\lambda''$ | I2 | $\delta\sigma$ | D | N | $\lambda$ | u(0.5,0.5) | $\dot{\lambda}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2.9E-01 | 0 | -3.3E+00 | 0 | 8.8E-02 | 0 | 1 | 7.9803556E+00 | 2.2727977E+00 | -6.2E-04 |
| 2 | -6.5E-04 | 0 | -3.3E+00 | 0 | -2.0E-04 | 0 | 0 | 7.9803557E+00 | 2.2723642E+00 | 3.6E-08 |
| 3 | 3.8E-08 | 0 | -3.3E+00 | 0 | 1.2E-08 | 0 | 0 | 7.9803557E+00 | 2.2723642E+00 | -1.4E-07 |

**Table 7-3:** Results for $F_2$, initial guess $\lambda_1 = 7.94617$, true Newton

| I | $\lambda'$ | I1 | $\lambda''$ | I2 | $\delta\sigma$ | D | N | $\lambda$ | u(0.5,0.5) | $\dot{\lambda}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4.7E-01 | 0 | -2.8E+00 | 0 | 1.6E-01 | 0 | 1 | 7.9791579E+00 | 2.3324510E+00 | -8.2E-02 |
| 2 | -9.7E-02 | 0 | -4.0E+00 | 0 | -2.4E-02 | 0 | 1 | 7.9803553E+00 | 2.2735657E+00 | -1.7E-03 |
| 3 | -1.9E-03 | 0 | -3.8E+00 | 0 | -5.1E-04 | 0 | 0 | 7.9803558E+00 | 2.2723647E+00 | -7.5E-07 |
| 4 | -8.5E-07 | 0 | -3.8E+00 | 0 | -2.2E-07 | 0 | 0 | 7.9803558E+00 | 2.2723642E+00 | -1.6E-07 |

**Table 7-4:** Results for $F_2$, initial guess $\lambda_2 = 7.5$, true Newton

| I | $\lambda'$ | I1 | $\lambda''$ | I2 | $\delta\sigma$ | D | N | $\lambda$ | u(0.5,0.5) | $\dot{\lambda}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4.8E-01 | 0 | -1.9E-01 | 0 | 2.6E+00 | 0 | 3 | 7.8877703E+00 | 2.8753642E+00 | -4.9E-01 |
| 2 | -2.2E-01 | 0 | -2.1E-01 | 0 | -1.1E+00 | 0 | 2 | 7.9699776E+00 | 2.1055603E+00 | 2.6E-01 |
| 3 | 8.1E-02 | 0 | -3.2E-01 | 0 | 2.6E-01 | 0 | 1 | 7.9803556E+00 | 2.2724285E+00 | -9.2E-05 |
| 4 | -3.0E-05 | 0 | -3.2E-01 | 0 | -9.4E-05 | 0 | 0 | 7.9803556E+00 | 2.2723643E+00 | -1.6E-07 |
| 5 | -5.2E-08 | 0 | -3.2E-01 | 0 | -1.6E-07 | 0 | 0 | 7.9803556E+00 | 2.2723642E+00 | 9.3E-10 |

**Table 7-5:** Results for $F_1$, initial guess $\lambda = 6.8$, true Newton

| I | $\lambda'$ | I1 | $\lambda''$ | I2 | $\delta\sigma$ | D | N | $\lambda$ | u(0.5,0.5) | $\dot{\lambda}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4.5E-01 | 0 | -1.1E+01 | 0 | 4.1E-02 | 0 | 2 | 6.8062598E+00 | 1.4189429E+00 | -1.9E-01 |
| 2 | -2.5E-01 | 0 | -2.7E+01 | 0 | -9.2E-03 | 0 | 1 | 6.8074830E+00 | 1.3951085E+00 | -2.6E-02 |
| 3 | -2.9E-02 | 0 | -2.1E+01 | 0 | -1.4E-03 | 0 | 0 | 6.8075035E+00 | 1.3916595E+00 | -4.5E-04 |
| 4 | -5.1E-04 | 0 | -2.0E+01 | 0 | -2.5E-05 | 0 | 0 | 6.8075035E+00 | 1.3915978E+00 | -3.3E-07 |
| 5 | -3.6E-07 | 0 | -2.0E+01 | 0 | -1.8E-08 | 0 | 0 | 6.8075035E+00 | 1.3915977E+00 | -1.3E-07 |

We observe from these results that :

- The computed turning points are *accurate* to within machine precision.
- The convergence is *quadratic*.
- The number of *inner* Newton iterations *decreases* rapidly as the turning point is approached. In fact, as the turning point is approached, the predictor is often so good that *no* Newton iteration is needed to satisfy the convergence criteria.
- No *damped*-Newton step is taken.

When compared to the methods surveyed in [13], our method seems to be more efficient. For example, for the cases corresponding to Tables 7-2 and 7-3, all of the methods in [13] took 4 iterations or more, whereas our method has converged after 2 and 3 iterations respectively.

Next, we tested the **chord** version on $F_2$, with difference approximations for second derivatives, which is the most efficient and most general version. The results are presented in Table 7-6 and 7-7.

**Table 7-6:** Results for $F_2$, initial guess $\lambda_0 = 7.96754$, chord/FD

| I | $\lambda'$ | I1 | $\lambda''$ | I2 | $\delta\sigma$ | D | N | $\lambda$ | u(0.5,0.5) | $\dot{\lambda}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2.9E-01 | 0 | -3.3E+00 | 0 | 8.8E-02 | 0 | 1 | 7.9803587E+00 | 2.2727739E+00 | - |
| 2 | -6.2E-04 | 2 | -3.3E+00 | 3 | -1.9E-04 | 0 | 0 | 7.9803588E+00 | 2.2723626E+00 | - |
| 3 | -3.3E-07 | 1 | -3.3E+00 | 2 | -9.9E-08 | 0 | 0 | 7.9803588E+00 | 2.2723624E+00 | -9.1E-08 |

**Table 7-7:** Results for $F_2$, initial guess $\lambda_1 = 7.94617$, chord/FD

| I | $\lambda'$ | I1 | $\lambda''$ | I2 | $\delta\sigma$ | D | N | $\lambda$ | u(0.5,0.5) | $\dot{\lambda}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4.7E-01 | 0 | -2.8E+00 | 0 | 1.6E-01 | 0 | 5 | 7.9791640E+00 | 2.3324056E+00 | - |
| 2 | -9.7E-02 | 6 | -4.0E+00 | 6 | -2.4E-02 | 0 | 1 | 7.9803528E+00 | 2.2735747E+00 | - |
| 3 | -2.0E-03 | 4 | -3.8E+00 | 5 | -5.1E-04 | 0 | 0 | 7.9803533E+00 | 2.2723666E+00 | - |
| 4 | -7.2E-07 | 1 | -3.8E+00 | 3 | -1.9E-07 | 0 | 0 | 7.9803533E+00 | 2.2723661E+00 | -7.0E-07 |

These results show that the *outer* iteration is very similar to the results of the basic algorithm. The *inner* iteration took a little more iterations because of the chord-Newton strategy, but due to the more accurate predictor, the number of inner iterations also *decreases* rapidly as the turning point is approached. As expected, both the inner Newton iterations and the iterative improvement took more iterations when the starting guess ($\lambda_1$) is farther away from the turning point. But the total number

of solves is still reasonably small considering only *one* factorization was performed. Note also that the number of iterative improvement iterations is less for $[u',\lambda']$ (which have a better initial guess from a first order predictor) than for $[u'',\lambda'']$.

To test the robustness of the **damped** version, we applied the true-Newton version on $F_2$, starting at $\lambda_4 = 7.0$ . The results are given in Table 7-8. Notice that the starting point is quite far away from the turning point and, as a consequence, many damped-Newton steps had to be taken in the beginning. As the turning point is approached, however, no damping is needed and quadratic convergence is regained.

**Table 7-8:** Results for $F_2$, initial guess $\lambda_4 = 7.0$, true Newton

| I | $\lambda'$ | I1 | $\lambda''$ | I2 | $\delta\sigma$ | D | N | $\lambda$ | u(0.5,0.5) | $\dot{\lambda}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 9.8E-01 | 0 | -3.3E-02 | 0 | 9.3E-01 | 5 | 3 | 7.8777199E+00 | 1.7912264E+00 | 7.1E-01 |
| 2 | 8.4E-01 | 0 | -7.5E-01 | 0 | 1.2E-01 | 3 | 2 | 7.9665943E+00 | 2.0815067E+00 | 3.0E-01 |
| 3 | 6.1E-01 | 0 | -5.9E+00 | 0 | 2.5E-02 | 2 | 2 | 7.9790850E+00 | 2.2123320E+00 | 9.0E-02 |
| 4 | 3.0E-01 | 0 | -2.6E+01 | 0 | 1.2E-02 | 0 | 2 | 7.9794920E+00 | 2.3233010E+00 | -7.0E-02 |
| 5 | -4.9E-01 | 0 | -2.0E+02 | 0 | -2.5E-03 | 0 | 1 | 7.9802267E+00 | 2.2919466E+00 | -2.8E-02 |
| 6 | -1.5E-01 | 0 | -9.7E+01 | 0 | -1.5E-03 | 0 | 1 | 7.9803523E+00 | 2.2755943E+00 | -4.6E-03 |
| 7 | -2.2E-02 | 0 | -7.1E+01 | 0 | -3.1E-04 | 0 | 0 | 7.9803558E+00 | 2.2724587E+00 | -1.4E-04 |
| 8 | -6.4E-04 | 0 | -6.7E+01 | 0 | -9.6E-06 | 0 | 0 | 7.9803558E+00 | 2.2723642E+00 | -1.6E-07 |

The next test we did was designed to show the effectiveness of the more accurate **second order predictor.** We repeated exactly the case corresponding to Table 7-4, except that the first order predictor was used instead. The result are presented in Table 7-9. They are very similar to the results in Table 7-4, except as expected, the number of inner Newton iterations does not decrease as the turning point is approached. Comparing the two tables shows the dramatic increase in efficiency made possible by the more accurate predictor.

The last test we performed was designed to test the effect of the **deflation techniques** used in Algorithm DBE, by running some tests using Algorithm BE instead. Without going into details, we shall just report that Algorithm BE is fairly reliable in practice, producing results that are practically the same as if Algorithm DBE had been used. A plausible explanation for the unexpected reliability of Algorithm BE is that it only fails when $G_u$ is *very* singular, at which point the accuracy is usually high enough that the iterations can be terminated. The only kind of problems that we have

**Table 7-9:** Results for $F_2$, initial guess $\lambda_2 = 7.5$, true Newton, 1st order predictor

| I | $\lambda'$ | I1 | $\lambda''$ | I2 | $\delta\sigma$ | D | N | $\lambda$ | u(0.5,0.5) | $\dot{\lambda}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 4.8E-01 | 0 | -1.9E-01 | 0 | 2.6E+00 | 0 | 4 | 7.8877699E+00 | 2.8753644E+00 | -4.9E-01 |
| 2 | -2.2E-01 | 0 | -2.1E-01 | 0 | -1.1E+00 | 0 | 4 | 7.9699775E+00 | 2.1055603E+00 | 2.6E-01 |
| 3 | 8.1E-02 | 0 | -3.2E-01 | 0 | 2.6E-01 | 0 | 4 | 7.9803549E+00 | 2.2724289E+00 | -9.2E-05 |
| 4 | -3.0E-05 | 0 | -3.2E-01 | 0 | -9.4E-05 | 0 | 4 | 7.9803550E+00 | 2.2723647E+00 | -1.6E-07 |
| 5 | -5.2E-08 | 0 | -3.2E-01 | 0 | -1.6E-07 | 0 | 4 | 7.9803548E+00 | 2.2723647E+00 | 3.6E-08 |

encountered with Algorithm BE is when an iterate happens to be *very* close to the turning point, then ||G|| can actually *increase* in the inner Newton iteration, causing a damped-Newton step to be taken. On the other hand, we have had no problem with Algorithm DBE at all, and we believe that it is to be preferred because of its higher reliability and minimal extra cost.

## 8. Conclusions

We have presented and tested a new algorithm for computing simple turning points of nonlinear equations. It possesses *quadratic convergence*, which, together with the more *accurate second order predictor*, makes it extremely fast when applied close to a turning point. We have also demonstrated that, through the use of a *chord*-Newton variant, the efficiency can be increased dramatically in such cases. On the other hand, when started far away from a turning point, its use of a natural *damped*-Newton strategy makes it reasonably *reliable and robust*. The use of the *block-elimination* algorithm with *implicit deflation* makes it possible to exploit special structures and solvers for the problem. Although *second derivatives* of G are required, the experimental results show that *difference approximations* for them can be used safely. Although more tests on different and larger problems are needed to more completely validate the new algorithm, our limited experimental results show rather convincingly that it is both *efficient* and *reliable* and compares favourably with the best of the existing methods.

# References


[1] J.P. Abbott.
An Efficient Algorithm for the Determination of Certain Bifurcation Points.
*Journal of Computational and Applied Mathematics* 4 :19 - 27, 1978.

[2] E. Allgower and K. Georg.
Simplicial and Continuation Methods for Approximating Fixed Points and Solutions to Systems of Equations.
*SIAM Review* 22(1):28 - 85, 1980.

[3] T.F. Chan.
*Deflation Techniques and Block-Elimination Algorithms for Solving Bordered Singular Systems.*
Technical Report 226, Yale Computer Science Department, New Haven, CT06520, 1982.

[4] T.F. Chan.
*Deflated Decomposition of Solutions of Nearly Singular Systems.*
Technical Report 225, Yale Computer Science Department, New Haven, CT06520, 1982.

[5] T.F. Chan and H.B. Keller.
Arclength Continuation and Multi-Grid Techniques for Nonlinear Eigenvalue Problems.
1982.
to appear in SIAM J. Sci. Stat. Comp., June, 1982.

[6] T.F. Chan.
*On the Existence and Computation of LU-factorizations with Small Pivots.*
Technical Report 227, Yale Computer Science Department, New Haven, CT06520, 1982.

[7] T.F. Chan and Y. Saad.
Iterative Methods for Solving Bordered Systems.
1982.
in preparation.

[8] J.J. Dongarra, J.R. Bunch, C.B. Moler and G.W. Stewart.
*LINPACK User's Guide.*
SIAM, Philadelphia, 1979.

[9] P.E. Gill, W. Murray, M.A. Saunders and M.H. Wright.
*A Procedure for Computing Forward-Difference Intervals for Numerical Optimization.*
Technical Report SOL 81-25, Systems Optimization Lab., Dept. of Operations Research, Stanford University, Stanford, 1981.

[10] H.B.Keller.
Numerical Solution of Bifurcation and Nonlinear Eigenvalue Problems.
In P. Rabinowitz (editor), *Applications of Bifurcation Theory*, pages 359-384. Academic Press, New York, 1977.

[11] H.B. Keller.
Global Homotopies and Newton Methods.
In Carl de Boor and Gene Golub (editor), *Recent Advances in Numerical Analysis*, pages 73-94. Academic Press, New York, 1978.

[12] M. Kubicek.
Dependence of Solution of Nonlinear Systems on a Parameter.
*ACM-TOMS* 2:98-107, 1976.

[13] R.G. Melhem and W.C. Rheinboldt.
*A Comparison of Methods for Determining Turning Points of Nonlinear Equations.*
Technical Report ICMA-82-32, Institute for Computational Mathematics and Applications, Department of Mathematics and Statistics, University of Pittsburg, Pittsburg, 1982.

[14] H.D. Mittelmann and H. Weber.
Numerical Methods for Bifurcation Problems - A Survey and Classification.
In H.D. Mittelmann and H. Weber (editors), *Bifurcation Problems and their Numerical Solution*, pages 1-45. Workshop on Bifurcation Problems and their Numerical Solution, January 15-17, Birkhauser, Dortmund, 1980.

[15] G. Moore and A. Spence.
The Calculation of Turning Points of Nonlinear Equations.
*SIAM J. Numer. Anal.* 17:567-576, 1980.

[16] J.C. Paumier.
Une Methode Numerique pour ke Calcul des Points de Retournement. Application a un Probleme aux Limites Non-lineaire.
*Numer. Math.* 37:433-444, 1981.

[17] G. Pönisch and H. Schwetlick.
Computing Turning Points of Curves Implicitly Defined by by Nonlinear Equations Depending on a Parameter.
*Computing* 26:107-121, 1981.

[18] W.C. Rheinboldt.
Solution Fields of Nonlinear Equations and Continuation Methods.
*SIAM J. Numer. Anal.* 17:221-237, 1980.

[19] W.C. Rheinboldt and J.V.Burkardt.
*A Program for a Locally-Parametrized Continuation Process.*
Technical Report ICMA-81-30, Institute for Computational Mathematics and Applications, Department of Mathematics and Statistics, University of Pittsburg, Pittsburg, 1981.

[20] W.C. Rheinboldt.
*Computation of Critical Boundaries on Equilibrium Manifolds.*
Technical Report ICMA-81-20, Institute for Computational Mathematics and Applications, Department of Mathematics and Statistics, University of Pittsburg, Pittsburg, 1980.

[21] W.C. Rheinboldt.
Numerical Analysis of Continuation Methods for Nonlinear Structural Problems.
*Computers and Structures* 13:103-113, 1981.

[22] H. Schwetlick.
*Effective Methods for Computing Turning Points of Curves Implicitly Defined by Nonlinear Equations.*
Technical Report 46, Martin Luther Univ., Halle Wittenberg, Sektion Mathem., , 1981.

[23] R. Seydel.
Numerical Computation of Branch Points in Nonlinear Equations.
*Numer. Math.* 33:339-352, 1979.

[24] R.B. Simpson.
A Method for the Numerical Determination of Bifurcation States of Nonlinear Systems of Equations.
*SIAM J. Numer. Anal.* 12(3):439-451, 1975.

[25] G.W. Stewart.
On the Implicit Deflation of Nearly Singular Systems of Linear Equations.
*SIAM J. Sci. Stat. Comp.* 2(2):136-140, 1981.