SIAM J. SCI. STAT. COMPUT.
Vol. 12, No. 3, p. 695–723, May 1991

012

VARIABLE ORDER ADAMS–BASHFORTH PREDICTORS WITH AN ERROR-STEPSIZE CONTROL FOR CONTINUATION METHODS*

BRUCE N. LUNDBERG† AND AUBREY B. POORE‡

Abstract. Variable order Adams–Bashforth predictors with an error-stepsize control are developed and used for a class of predictor-corrector continuation methods which trace solution paths of an underdetermined nonlinear system $F(w) = 0$, where $F : \mathbb{R}^{n+1} \to \mathbb{R}^n$. The predictor method developed here employs consecutive order Adams–Bashforth predictors to determine the stepsize by estimating the local truncation error in the lower order formula, a comparison of three consecutive order methods to choose the order that yields the largest stepsize, and local extrapolation to improve accuracy. Unlike many differential equation algorithms, the stepsize is changed at each step to give more uniformity in the predictor error and the global error is controlled at each step through the use of a Newton-like correction back to the path after each prediction. The philosophy of following the path closely with only one or two corrections per step is shown to yield an efficient and robust algorithm which is fairly insensitive to parameter settings. The key to the success and robustness of these multistep predictors is the approximation of arclength along the path at an accuracy commensurate with that of the tangents to the path, which is achieved by a local parameterization suggested by the corrector equations. To demonstrate the efficiency and robustness of the resulting continuation algorithm, several numerical comparisons are made with state-of-the-art software packages HOMPACK and PITCON.

Key words. predictor-corrector continuation, continuation, Adams–Bashforth predictors, error-stepsize control, Davidenko equation, arclength parameterization

AMS(MOS) subject classifications. 65H10, 34A50, 65L05

1. Introduction.

Predictor-corrector methods for tracing solution paths of an underdetermined system of nonlinear equations $F(w) = 0$, where $F : \mathbb{R}^{n+1} \to \mathbb{R}^n$, have proved to be effective numerical procedures for various problems ranging from homotopy methods for nonlinear equations and fixed point problems to parametric investigations of dynamical systems and nonlinear eigenvalue problems. These methods are now in wide use and are considered to be reasonably robust but slow and computationally intensive, primarily due to the extensive linear algebra requirements in both the predictor and corrector phases. To describe these procedures, we assume that $F(w)$ is continuously differentiable, that $F(w) = 0$ has a smooth solution path $P = \{w \in \mathbb{R}^{n+1} : w = \Psi(t), t \in I\}$, where I is an interval of real numbers, and that the path is nonsingular in that $\text{rank}[D_w F|_{w \in P}] = n$. Most path following algorithms generate a sequence of points $\{(t_k, w_k)\}_{k=0}^N$, where w_k is on or near the path and w_0 is a known solution of $F(w) = 0$. To go from a point w_k to a point w_{k+1}, we first predict a new point, say, wp_{k+1}, using past and present information, and then correct back to the path using a Newton or Newton-like method. Besides these prediction and correction phases, there is usually an error-stepsize control to improve efficiency and a method for computing special points on the path such as singularities or, in case of

*Received by the editors December 27, 1988; accepted for publication (in revised form) March 12, 1990. This work was partially supported by National Science Foundation grant DMS-87-04679 and by Air Force Office of Scientific Research grant AFOSR-88-0059.

†Department of Mathematics, Grand Canyon University, 3300 W. Camelback, Phoenix, Arizona 85017. The work of the first author was also partially supported by National Aeronautics and Space Administration grant NGT-06-002-802.

‡Department of Mathematics, Colorado State University, Fort Collins, Colorado 80523.

a homotopy problem, the solution corresponding to a specific value of the homotopy parameter. Of course, the subject can become more complex when branching switching [11], [14], [16], [19], [33] at singularities and the ability to follow singularities in parameter space are added; however, the concentration in this work is on the predictor phase and an appropriate error-stepsize control with the goal being to add efficiency and robustness to these methodologies.

Several suggestions have been made for the prediction phase. Certainly we can combine points on the curve with polynomial or rational function extrapolation to produce wp_{k+1}. The use of ordinary differential equation techniques arises from the observation that dw/dt formally satisfies $[D_w F(w)]dw/dt = 0$, which is obtained by differentiating $F(w(t)) = 0$ with respect to t. Thus the solution $w = \Psi(t)$ locally satisfies the initial value problem

$$\frac{dw}{dt} = T(w), \qquad w(t_0) = w_0, \tag{1.1}$$

where $F(w_0) = 0$ and $T(w)$ is a properly oriented nonzero null vector of $D_w F(w)$ which can be computed in various ways [1], [16], [19], [30], [42]. (A rigorous justification of a local solution of (1.1) can be based on the assumed smoothness of $F(w)$, the full rank condition above, and the implicit function theorem.) The scaling of $T(w)$ depends on the choice of the parameterization of P; for example, if $||T(w)||_2 = 1$, then t is arclength. (The celebrated Davidenko differential equation $D_u F(u,\lambda)du/d\lambda + D_\lambda F(u,\lambda) = 0$ is obtained by formally differentiating $F(u,\lambda) = 0$ with respect to λ and is a special case of (1.1) in which $w^T = (u^T, \lambda)$ and λ is the independent parameter. A tangent in this case is $T(w)^T = \pm(du/d\lambda^T, 1)$.) Another possibility is to combine solution values w and tangent values $T(w)$ in an extrapolation technique such as Hermite extrapolation. Each of these possibilities has been suggested and investigated in the literature [5], [12], [24], [32], [44], [47], and more recently, Taylor and Padé predictors based on finite-difference approximation of higher derivatives have been used effectively by Schwetlick and Cleve [36]. However, Allgower and Georg [1], [14], [15] recommend the use of Adams–Bashforth predictors since they do not demand the computation of higher derivatives and are *usually* less sensitive to variations in the accuracy of the computed solution values, w, than are methods which involve the solution values in the extrapolation process. (The example given in the next paragraph shows that this need not always be the case.) From the various techniques for initial value problems arising in ordinary differential equations, we generally recommend the use of a multistep method when function evaluations are extremely expensive. We were thus motivated to investigate these Adams–Bashforth predictors.

The design philosophies of error-stepsize control for curve following algorithms are also many and varied. At one end of the spectrum is the actual use of computer software for initial value problems [3], [14], [45] to solve the differential equation (1.1). The difficulty with this as a general continuation procedure is that the solution of the initial value problem (1.1) or the Davidenko equation may be stable or unstable and the equation may even be stiff. An artificial but simple illustration of this is given by the problem

$$F(x,y,\lambda) = \begin{pmatrix} xe^{a\lambda} \\ ye^{b\lambda} \end{pmatrix} = 0,$$

whose corresponding Davidenko equation is

$$\frac{d}{d\lambda}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -a & 0 \\ 0 & -b \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}.$$

This equation can be stiff or nonstiff and the solution $(x, y) = (0, 0)$ may be stable or unstable, depending on the relative magnitudes and signs of the parameters a and b. Watson [45, p. 305] has, however, shown that under certain hypotheses the use of initial value techniques for ordinary differential equations yields an algorithm that is "inherently stable near the fixed point" for a class of probability one homotopy mappings used for the solution of fixed point problems. For other problem classes, particularly natural parameter continuation, curve following over any distance without using extreme error tolerances can lead to highly inaccurate solutions, and thus the stepsizes are required to be relatively small to maintain a reasonable global error. The use of a Newton correction back to the path P after a prediction resolves the global error problem, but the small stepsize problem still remains. At the other end of the spectrum are those methods which employ a low-order predictor with a stepsize chosen to be as large as possible while giving a prediction within the domain of convergence of Newton's method. This is an appealing philosophy, but it is not without its deficiencies: the number of iterations in the corrector process may be large, the actual estimation of the domain of convergence of Newton's method must itself be extrapolated, and robustness can be sacrificed due to curve jumping. Indeed, we have found that we often need to make several runs at different error tolerances to achieve success in continuation codes, especially when we are trying to achieve efficiency. Thus the question is whether a predictor-corrector method can be designed wherein the predicted point remains "close" to the curve for robustness and yet the efficiency of the methodology is maintained or increased by reducing the amount of work required to solve a given problem to a specified accuracy.

In this work, the Adams–Bashforth predictors are investigated with the objective being to develop a sufficiently accurate predictor so that only one or two corrector iterations back to the curve are required at each step. The error-stepsize control developed here employs consecutive order Adams–Bashforth predictors to determine the stepsize by estimating the local truncation error in the lower order formula, a comparison of three consecutive order methods to choose the order that yields the largest stepsize, and local extrapolation to improve accuracy. This methodology differs significantly from the Adams methods used in the numerical solution of ordinary differential equations, including the Adams predictors used in HOMPACK [42], [44]. Specifically, the stepsize is changed at each step to give more uniformity in the predictor error and larger stepsizes. The error-stepsize control and order selection are based on consecutive order Adams–Bashforth (explicit) formulas rather than the Adams–Moulton corrector (implicit) formulas. (As will be discussed further in §4, this is a fundamental difference with the ordinary differential equations methodology.) Finally, the error controlled at each step is the actual (global) error rather than the local error or error per unit step. This allows the use of larger error tolerances, which also results in larger stepsizes. The key, however, to the success and robustness of these multistep predictors is the approximation of arclength along the path at an accuracy commensurate with that of the tangents to the path, and this is achieved by a local parameterization suggested by the corrector equations.

In the design of a predictor-corrector continuation procedure there is a delicate balance between robustness, which is generally achieved by following the path closely, and the goal of minimizing the amount of work required to solve a given problem to a specified accuracy. To minimize the work, one can design the procedure to minimize the number of matrix factorizations, the number of function evaluations, the number of linear systems solved, or the overall work (cpu seconds) required. This design is, however, problem size dependent. On medium scale dense problems the number of

matrix factorizations may well dominate the overall work required, while on very small problems the overhead, other than the solution of linear systems, may dominate the computations. If the system has large sparse Jacobians and iterative methods are used to solve the linear systems, then the number of linear systems solved may be an appropriate measure of the work involved. Furthermore, the problem objective may differ depending on the usage. We may wish to follow a homotopy path wherein the goal is to traverse the path as cheaply as possible and to compute the solution on the path at a special value of the homotopy parameter. In this case the path itself is of little interest and only serves as a safety net. (This problem class is sometimes called artificial parameter continuation [45].) Another class of problems is the natural parameter continuation that arises, for example, in nonlinear eigenvalue problems and parametric studies of dynamical systems. Here the details of the curve itself are of the utmost importance in that the objective is to detect changes in structure and number of the solutions over a parameter range. Thus the design objective is not a simple issue and we report the operational counts for several of these performance measures in §7. In reporting computational results for comparative purposes, we have also found that the required parameter settings in such state-of-the-art software as HOMPACK [44] and PITCON [30], [31], [33], as well as our own ABCON, are not always clear. Thus we may need to make several runs on a given problem before it succeeds in computing the final answer properly or in computing the details of the path. This raises the issue of whether one should report only the most successful run or some measure of the total effort required to "solve" a problem. We have chosen to do both by reporting, in §7, the number of runs made to obtain success and statistics for this successful run. (We actually search for the largest tolerances which lead to success.) Finally, we mention three references in which the minimum work as measured by the number of Jacobian factorizations (Newton steps) [35], [40] or the number of linear systems solved [33] was the design criterion. In the first the author himself claims the results are nonconstructive; the second applies only to homotopy methods via classical continuation (correction with λ fixed) and requires knowledge of global parameters which must be estimated in some way; in the latter, PITCON Full was designed to minimize the number of linear systems solved and indeed the code works quite well in this aspect. In our opinion, both theoretical and practical implementation issues remain open in the trade-offs between robustness and minimum work requirements.

The remaining sections of this paper are organized as follows: In §2 the formulas for the Adams–Bashforth predictors are summarized. The computation of arclength along the path is developed in §3, and the error-stepsize control and order selection is developed in §4. Section 5 contains a discussion of the correction phase, methods for dealing with corrector failures, and the computation of the tangent. These procedures are summarized in a high-level pseudocode in §6. To give a comparison of the efficiency and robustness of these Adams–Bashforth predictors, several test problems from the literature are solved and the results are compared with those of HOMPACK [44] and PITCON [31] in §7.

2. The Adams–Bashforth predictors. For completeness, we state in this section the Adams–Bashforth formulas for the initial value problem (1.1). These variable stepsize formulas are due to Krogh [20] and have particularly good roundoff error properties [37]. With the exception of a shifted index these formulas can be found in Shampine and Gordon [37].

Let $P = \{w \in \Re^{n+1} : w = \Psi(t), t \in I\}$, where I is an interval of real numbers,

denote a parameterization of the solution path of $F(w) = 0$ which is assumed to be nonsingular in that $[D_w F|_{w \in P}]$ has full rank on the path P. An integration of the differential equation (1.1) yields

$$w(t_k + \Delta t) = w_k + \int_{t_k}^{t_k + \Delta t} T(w(\sigma)) d\sigma \tag{2.1}$$

Assume $w_i = w(t_i)$ and $T(t_i) = T(w(t_i))$ have been computed for $i = 0, 1, \cdots, k$, and let $P_{k,m}(t)$ be the vector of polynomials, of degree not exceeding m, which interpolates T at $t_k, \cdots, t_{k-m}$. Employing the backward Newton form for this interpolating polynomial, we have

$$T(w(t)) = P_{k,m}(t) + \Gamma_{k,m}(t) T[t_k, \cdots, t_{k-m}, t], \tag{2.2}$$

where $P_{k,m}(t) = \sum_{i=0}^{m} \{\Gamma_{k,i-1}(t) T[t_k, \cdots, t_{k-i}]\}, \Gamma_{k,i}(t) = \prod_{j=0}^{i} (t - t_{k-j})$ for $i \geq 0, \Gamma_{k,-1}(t) = 1$, and $T[t_k, \cdots, t_{k-m}]$ denotes the (vector) Newton divided difference [37], [38]. Equations (2.1) and (2.2) yield

$$w(t_k + \Delta t) = w_k + \Delta t d_{k,m}(\Delta t) + E_{k,m}(\Delta t), \tag{2.3}$$

where

$$\begin{aligned} d_{k,m}(\Delta t) &= \frac{1}{\Delta t} \int_{t_k}^{t_k + \Delta t} P_{k,m}(\sigma) d\sigma, \\ E_{k,m}(\Delta t) &= \int_{t_k}^{t_k + \Delta t} \Gamma_{k,m}(\sigma) T[t_k, \cdots, t_{k-m}, \sigma] d\sigma. \end{aligned} \tag{2.4}$$

The predictor of $w(t_k + \Delta t)$ is taken to be

$$wp(k+1, m, \Delta t) = w_k + \Delta t d_{k,m}(\Delta t), \tag{2.5}$$

which has an error $E_{k,m}(\Delta t)$. Using the following notation

$$\begin{aligned} &\psi_{k,i} = \sum_{j=0}^{i} \Delta t_{k-j} = t_{k+1} - t_{k-i}, \quad \text{where } \Delta t_k = t_{k+1} - t_k = \Delta t \\ &\beta_{k,i} = \begin{cases} 1, & i = 0, \\ \prod_{j=0}^{i-1} \frac{\psi_{k,j}}{\psi_{k-1,j}}, & i \geq 1, \end{cases} \\ &\alpha_{k,i} = \Delta t_k / \psi_{k,i} \quad \text{and} \quad \phi_{k,i} = \prod_{j=0}^{i-1} \psi_{k-1,j} T[t_k, \cdots, t_{k-i}], \end{aligned} \tag{2.6}$$

the direction $d_{k,m}(\Delta t)$ can be expressed as

$$d_{k,m}(\Delta t) = \sum_{i=0}^{m} \phi_{k,i} \beta_{k,i} g_{i1}, \tag{2.7}$$

and the predicted tangent at w_{k+1} as

$$Tp_{k+1,m}(\Delta t) = P_{k,m}(t_k + \Delta t) = \sum_{i=0}^{m} \phi_{k,i} \beta_{k,i}, \tag{2.8}$$

where $g_{i1}, \phi_{k,i}$, and $\beta_{k,i}$ can be efficiently computed from the following recurrence relationships:

$$
\begin{aligned}
& g_{0j} = \frac{1}{j} \quad \text{and} \quad g_{1j} = \frac{1}{[j \cdot (j+1)]}, \\
& g_{ij} = g_{i-1,j} - \alpha_{k,i-1} g_{i-1,j+1}, \qquad \begin{pmatrix} i = 2, \cdots, m \\ j = 1, \cdots, m-i+1 \end{pmatrix}, \\
& \phi_{k+1,i+1} = \phi_{k+1,i} - \beta_{k,i} \phi_{k,i} \quad (i = 0, \cdots, m) \quad \text{where } \phi_{k+1,0} = T(t_{k+1}), \\
& \psi_{k,i} = \psi_{k-1,i-1} + \triangle t, \qquad i \geq 1, \\
& \beta_{k,i} = \beta_{k,i-1} \frac{\psi_{k,i-1}}{\psi_{k-1,i-1}}, \qquad i \geq 1.
\end{aligned} \tag{2.9}
$$

3. The local and global parameterizations of the path: The estimation of arclength. The use of arclength as a global continuation parameter is recommended and used by many authors [2], [16], [17], [21], [22], [25], [26], [42], [44], especially as a way of dealing with limit or fold point singularities. In some of these a local parameterization such as Keller's pseudo-arclength [16] is actually used instead of arclength itself. Watson, Billups, and Morgan [44], [45], on the other hand, use arclength as the actual independent parameter in the ordinary differential equation method HOMPACK (DF and DS), but this is the arclength of the path tracked by the solution of the differential equation (1.1) versus true solution path as defined by the equation $F(w) = 0$. (This particular method does not perform a Newton-like correction back to solution path of $F(w) = 0$, but is appropriate for his intended problem class.) Arclength has also been suggested as an adaptive mesh selection technique for boundary value problems for ordinary differential equations [34] in that a uniform meshsize in arclength tends to place the needed meshpoints "appropriately" along an interval of interest. While the computation of the stepsizes in arclength may be too expensive for a general mesh selection scheme for boundary value problems, its use here is warranted in view of the extreme expense of the function evaluations, which requires the solution of several linear systems. The use of arclength along the solution path tends to give stepsizes that are more or less equal in arclength except in extremely sharp turns, where we generally reduce the multistep method to Euler's method. These more or less equal stepsizes add robustness to the predictors and the error-stepsize control as well as efficiency to the methodology.

When a Newton or Newton-like correction to the true path, defined by $F(w) = 0$, is used in the corrector phase and when a multistep method involving more than two steps is used as a predictor, an accurate estimation of arclength along the solution path of $F(w) = 0$ is required. In the absence of this accurate approximation we have found that Hermite and Adams–Bashforth multistep predictors with more that two steps degenerate severely in that the stepsizes become "too small" to be efficient. This more accurate arclength computation is not needed in either HOMPACK or PITCON: the ODE method in HOMPACK (DF and DS) does not involve a Newton-like correction back to the solution of $F(w) = 0$; at most two-step Hermite predictors are used in the other two methods in HOMPACK; and Rheinboldt's PITCON uses an Euler predictor. Thus the key to the efficiency of the Adams–Bashforth predictors, as developed in this work, is the estimation of arclength along the solution path at an accuracy commensurate with that of the predictors. The next goal is to explain how this more accurate approximation is made. In this derivation, the letter "t" will continue to be used as the independent variable in the generic parameterization for

(1.1), the letter "s" will denote arclength along the path P, and "h" will be a local variable as explained later.

Once a prediction $wp(k+1,m,h_k) = w_k + h_k d_{k,m}(h_k)$ is made, the corrected point, w_{k+1}, on the solution path is obtained by solving an augmented system of equations

$$\begin{bmatrix} F(w) \\ N(w) \end{bmatrix} = 0, \tag{3.1}$$

where $N : \mathbb{R}^{n+1} \to \mathbb{R}$.

The predictor steplength, h_k, must also be "corrected" to Δs_k and this depends on the particular augmenting equation $N(w)$. Three important examples are

$$N_1(w) = (w - wp(k+1,m,h))^T d_{k,m}(h), \tag{3.2}$$

which confines the correction to a hyperplane orthogonal to the predictor direction $d_{k,m}(h)$,

$$N_2(w) = \|w - w_k\|_2^2 - h^2 \|d_{k,m}(h)\|_2^2, \tag{3.3}$$

which confines the correction to a sphere centered at w_k, and, for $w^T = (u^T, \lambda)$ with the parameter being in the $(n+1)$st position

$$N_3(w) = e_{n+1}^T (w - wp(k+1,m,h)), \tag{3.4}$$

which corresponds to the classical correction with the parameter λ being held fixed during the correction process. Equation (3.2) corresponds to tangent [16], (3.3) to secant [25], and (3.4) to natural (λ) parameterizations of the solution path. The first two schemes allow effective computation around fold (limit, turning) point singularities and define a local parameterization [4] of the curve in the following way: given h, compute $wp(k+1,m,h)$ and then correct back to the curve to obtain a locally unique solution $w(h)$. (Some correction processes, such as Georg's normal flow algorithm [1], [2], [14], do not fit this characterization.) The following propositions make these statements precise and allow effective computation of arclength along the solution path P. In the first theorem functional and derivative information needed in the computation of arclength is established, while the next two theorems establish existence and certain properties of the local parameterization for the augmenting equation $N_1(w)$; however, similar results hold for $N_2(w)$. Since the proof of Theorem 1 follows by direct computation and the other two theorems follow easily from the implicit function theorem [9], the proofs are omitted.

THEOREM 1. *Let $F : \mathbb{R}^{n+1} \to \mathbb{R}^n$ be continuously differentiable in an open neighborhood of $P = \{w \in \mathbb{R}^{n+1} : w = \Psi(t), F(\Psi(t)) = 0, t \in I\}$, where I is an interval of real numbers and assume that the path P is nonsingular in that $[D_w F|_{w \in P}]$ is of full rank. Let $T_k = T(w(t_k))$, where $T(w(t))$ denotes the tangent vector $dw/dt = \Psi'(t)$, and extend the definition of $d_{k,m}(h)$ given in (2.4) to*

$$d_{k,m}(h) = \begin{bmatrix} h^{-1} \int_0^h P_{k,m}(t_k + \sigma) d\sigma, & h \neq 0 \\ T_k, & h = 0 \end{bmatrix}.$$

Then $d_{k,m}$ is a polynomial in h and

$$d'_{k,m}(h) = \begin{bmatrix} (P_{k,m}(t_k + h) - d_{k,m}(h))/h, & h \neq 0 \\ T[t_k, t_{k-1}]/2 & h = 0 \end{bmatrix}.$$

In the next theorem, the existence and uniqueness of the local parameterization of the curve is stated for completeness. This local versus global parameterization is certainly implied in, for example, the work of Allgower and Georg [1], [2]; Keller [16], [17]; Rheinboldt and Burkardt [30], [33]; and Watson [45].

THEOREM 2. *Let $F : \mathbb{R}^{n+1} \to \mathbb{R}^n$ be at least C^1 in a neighborhood B of w_k, where w_k is a nonsingular point on the path P defined in Theorem 1, and let T_k be a unit vector in the null space of $D_w F(w_k)$. Consider $G : \mathbb{R}^{n+2} \to \mathbb{R}^{n+1}$ as given in* (3.1)–(3.2), *namely,*

$$G(w,h) = \begin{bmatrix} F(w) \\ d_{k,m}(h)^T(w - w_k - h d_{k,m}(h)) \end{bmatrix},$$

where $d_{k,m}$ is defined as in Theorem 1. *Then the equation $G(w,h) = 0$ defines a local parameterization of the path P by h for h in a sufficiently small neighborhood of zero; that is, there exist (open) neighborhoods $I = (-\epsilon, \epsilon)$ and $U \subset B$ of w_k such that for each $h \in I$ there is a unique $w \in U$ for which $G(w,h) = 0$ and dw/dh is continuous and nonzero. Furthermore, the function $w = w(h)$ is as smooth as F on I, and satisfies $dw(0)/dh = T_k$.*

THEOREM 3. *Let F satisfy the assumptions of Theorem* 1, *s denote the arclength along the solution path P with $w_k = w(s_k)$, h be the parameter given in Theorem* 2, *and $T(w)$ be a unit vector in the nullspace of $D_w F(w)$ with orientation corresponding to the parameter s. Then there is a neighborhood $I = (-\epsilon, \epsilon)$ of $h = 0$, and a corresponding neighborhood V of $s = s_k$ such that $s(h) : I \to V$ is a well-defined, one-to-one function which is as smooth as F. Furthermore, $s = s(h)$ satisfies $ds(0)/dh = 1$ and*

$$\frac{ds}{dh}(h) = \frac{d_{k,m}(h)^T d_{k,m}(h) - [w(h) - w_k - 2h d_{k,m}(h)]^T d'_{k,m}(h)}{T(w(h))^T d_{k,m}(h)}. \tag{3.5}$$

We now assume that the path P is parameterized by arclength so that "s" will replace the variable "t" in the theorems above. Unfortunately, the local parameter h is not a sufficiently accurate approximation to the arclength along the curve from w_k to $w(h)$. Given a step h_k, better estimates of arclength can be obtained by approximating the arclength integral in the local parameter $\triangle s_k = \int_0^{h_k} \|dw(h)/dh\|_2 dh$. Note that since dw/ds is a unit vector and $dw/dh = (dw/ds)(ds/dh)$, the arclength increment $\triangle s_k$ may be computed as

$$\triangle s_k = \int_0^{h_k} \left| ds(h)/dh \right| dh. \tag{3.6}$$

Now $ds(h_k)/dh$ can be obtained cheaply from the formulas given in the above theorems since the quantities $h_k, d_{k,m}(h_k)$, and $P_{k,m}(s_k + h_k)$ have been computed for the predictor $wp(k+1, m, h_k)$, $w(h_k)$ is the new corrected point, and $T_{k+1} = T(w(h_k))$ must in any case be computed for the next predictor step. Thus, in the normal course of taking steps along the path, $ds(h)/dh$ is known at $h = 0$ and $h = h_k$. Simple integration rules using this information include the rectangle rule ($s_{k+1} - s_k = h_k(ds(0)/dh) + \mathcal{O}(h_k^2) = h_k + \mathcal{O}(h_k^2)$) and the trapezoidal rule ($s_{k+1} - s_k = .5h_k(ds(0)/dh + ds(h_k)/dh) + \mathcal{O}(h_k^3)$). These rules are, however, not sufficiently accurate for the higher-order predictors, so that we must turn to Gaussian or some other higher-order quadrature rules. The difficulty encountered in this direction is the prohibitively expensive cost of evaluating $ds(h)/dh = \|dw(h)/dh\|_2$ at values of h between zero and h_k: we must first predict

and then correct back to the curve to evaluate $ds(h)/dh$. One possibility is to approximate $\triangle s_k = \int_0^{h_k} \|dw(h)/dh\|_2 dh$ by extrapolating $dw(h)/dh$ from the polynomial $P_{k,m}(s_k + h)$ which interpolates $dw(h)/dh$ at $h = 0, -\triangle s_{k-1}, \cdots, -(s_k - s_{k-m})$. Since we must compute the $(k+1)$st point and unit tangent anyway, we prefer to use an updated polynomial and thus polynomial interpolation. After h_k and $dw(h_k)/dh$ are known, a new interpolating polynomial $\tilde{P}_{k+1,m+1}$ of degree $m+1$ can be constructed which is an update of $P_{k,m}$, incorporates the stepsize h_k and renormalized tangent $dw(h_k)/dh$, and agrees with $P_{k,m}$ at s_i for $i = k, \cdots, k-m$. Then we may write (assuming $s_i - s_{i-1} = \mathcal{O}(h_k)$ as $h_k \to 0$ for $i = k, \cdots, k-m+1$)

$$s_{k+1} - s_k = \int_0^{h_k} \left\| \frac{dw}{dh}(\xi) \right\|_2 d\xi = \int_0^{h_k} \left\| \tilde{P}_{k+1,m+1}(s_k + \xi) \right\|_2 d\xi + \mathcal{O}(h_k^{m+3}). \tag{3.7}$$

The latter integral may be estimated using a higher-order quadrature rule as discussed above. The order of the integration scheme used should be commensurate with the error in approximation (3.7). The Gaussian–Lobatto integration rules [39] are a good choice since they make explicit use of the exact information at the endpoints. Since this rule with q points has an asymptotic error which is $\mathcal{O}(h_k^{2q-1})$ [38], q is chosen to satisfy $2q - 1 > m + 3$ so that the error from numerical quadrature does not contribute (asymptotically) to the dominant error term in (3.7). The details of these computations are summarized in the following high-level pseudocode.

Arclength Correction Algorithm. Given the predicted tangent $T = Tp_{k+1,m}(h_k)$, new tangent T_{k+1}, point w_{k+1} on the path P, the predictor degree $m := m_k$ together with $\varphi_{k,i}$ and $\beta_{k,i}$ for $i = 0, \cdots, m$, the point w_k, the prediction direction $d = d_{k,m}(h_k)$, and the predictor stepsize h_k from the kth predictor step, the following algorithm produces an improved estimate of the stepsize $\triangle s_k$ in arclength between w_k and w_{k+1} on the path P:

1. $d' := (Tp - d)/h_k := (P_{k,m}(s_k + h_k) - d)/h_k$ { Theorem 1, §3, (2.8) }.
2. $ds(0)/dh := 1$ { Theorem 3, §3 }
3. $ds(h_k)/dh := [d^T d - (w_{k+1} - w_k - 2h_k d)^T d']/T_{k+1}^T d$ {Theorem 3, §3 }.

Compute modified divided differences for polynomial $\tilde{P}_{k+1,m+1}$:

4. $\tilde{\varphi}_{k+1,0} := (ds(h_k)/dh)T_{k+1}$.
5. Set $\tilde{\varphi}_{k+1,j} := \tilde{\varphi}_{k+1,j-1} - \beta_{k,j-1}\varphi_{k,j-1}$ for $j = 1, \cdots, m+1$ { Use (2.9)}
6. Choose the number of Lobatto points: $q :=$ integer part $(m+5)/2$ and let π_i and ω_i for $i = 0, \cdots, q-1$ denote the Lobatto points and corresponding weights on the interval [0,1] [39].
7. Perform the Lobatto quadrature on (3.7):

$$\triangle s_k := \omega_o \cdot \frac{ds}{dh}(0) + \sum_{i=1}^{q-2} \omega_i \|\tilde{P}_{k+1,m+1}(s_{k+1} + (1 - \pi_i)h_k)\|_2 + \omega_{q-1} \cdot \frac{ds}{dh}(h_k).$$

{ Here, the $\tilde{P}_{k+1,m+1}(s_k + (1-\pi_i)h_k) = Tp_{k+2,m+1}((1-\pi_i)h_k)$ are computed using formula (2.8) with $\Delta t := (1 - \pi_i)h_k$, the $\tilde{\varphi}$'s in place of the φ's, and m and k replaced by $m+1$ and $k+1$, respectively.}

4. Error-stepsize control and order selection. Given error tolerances for the predicted point, the error-stepsize control developed here yields an estimate of the stepsize that satisfies these error tolerances and then adjusts the order to produce the largest such stepsize. Although this methodology is motivated by similar techniques

in ordinary differential equations, there are significant differences that arise out of the objective to track a solution of the underdetermined system of equations $F(w) = 0$ rather than to solve the initial value problem (1.1). This is most evident in the correction process which involves a Newton or Newton-like correction (solve) back to the solution of the system $F(w) = 0$. Two fundamental implications of this correction are that the global rather than local error is controlled at each step and, due to the expense of the correction process, (it involves the solution of several linear systems), the error-stepsize control generally (but not always) avoids the corrector in the decision of the stepsize. Thus we use consecutive order Adams–Bashforth predictors to estimate the stepsize that should be taken in the present step rather than using the Adams–Bashforth–Moulton pair as in ordinary differential equations [37]. Also, in ordinary differential equations we frequently use as the current stepsize an optimal stepsize that should have been taken on the previous step, but this strategy is too expensive here since it involves the corrector process. Although this procedure is not fail-safe, the correction process and the underlying equation $F = 0$ provide a verification of the accuracy of the predictor. Thus we can be more aggressive in the choice of stepsize—if the stepsize is too large or the prediction is too inaccurate, we can detect this, modify the predictor by cutting the stepsize and order, and restart the corrector. Corrector failures are an inevitable consequence of this strategy but are acceptable in that we can deal with them effectively; however, too many such failures can cause the methodology to be inefficient. (We hasten to emphasize that corrector failures are generally due to sharp turns in the solution path which may or may not be associated with singularities on the path.) In the remainder of this section, we present the error-stepsize control and order selection scheme.

To estimate a stepsize $h_k := \gamma h_{k-1}$ which achieves a combined absolute (Eabs) and relative (Erel) error requirement in the predictor, we use consecutive order Adams–Bashforth formulas to estimate the error in the lower order formula. Requiring that the estimate of the error in each component meet the given tolerances gives the formula

$$\gamma_m = \min_i \left[\frac{\text{Eabs} + \text{Erel}|[wp(k+1, m+1, h_{k-1})](i)|}{2|[wp(k+1, m+1, h_{k-1}) - wp(k+1, m, h_{k-1})](i)|} \right]^{1/(m+2)}, \tag{4.1}$$

where the customary 2 has been added in the denominator inside the radical to account for the asymptotic nature of this formula and the (i) refers to the ith component of the vector contained in brackets. Our algorithm makes use of this estimate to select the order and stepsize at each step. In general, estimates γ_{m-1}, γ_m and γ_{m+1}, corresponding to three consecutive orders, are computed where m is the order of the predictor used on the previous step. The order giving the largest value of γ is selected, along with the corresponding stepsize γh_{k-1} subject to the constraints that $\gamma \in [0.1, 10.0]$, and $\gamma h_{k-1} \in [h_{\min}, h_{\max}] = [10^{-7}, 100.0]$. Then, assuming the stepsize has not been decreased, local extrapolation is used, that is, the predictor of order one higher than the selected order is used. (The use of local extrapolation followed by a Newton-like correction generally produces between a half and a full digit of additional accuracy in the corrected point on the curve.) Finally, we impose a maximum order $Maxm$, which we typically choose to be four when Eabs = Erel = 0.01 is used. For smaller predictor tolerances, larger maximum orders are suggested.

Asymptotic estimates similar to (4.1) with associated error-stepsize and order controls have been discussed in Shampine and Gordon [37] and Krogh [20], so we will not derive our version here. Krogh [20], however, points out several advantages to allowing even small changes in the stepsize at every step, among which is greater

efficiency in terms of function evaluations. In our cases the overhead involved in recomputing the integration coefficients at each step is insignificant in comparison to the cost of the linear algebra at each step, so we allow the stepsize to change on each step. Timing studies on a Cyber 205, using problems one and five in §7 demonstrated that the entire prediction phase of the algorithm typically consumes less than 10 percent of the CPU time for problems of dimension 20 and less than 1 percent on problems of dimension 100.

An examination of the error term (2.4) suggests several appropriate safeguards in the use of (4.1) [37]. The derivation of this formula assumes that $E_{k,m}(h_k) = \mathcal{O}(h_k^{m+2})$ as $h_k \to 0$. If $h_{k-i} = \mathcal{O}(h_k)$ for $i = 1, \cdots, m$ and w is sufficiently smooth, this is indeed the case. On the other hand, if $h_k << h_{k-i}$ for $i = 1, \cdots, m$ then $E_{k,m}(h_k) = \mathcal{O}(h_k^2)$, so that the predictor behaves like Euler's method. We can conclude that when the stepsize is being drastically reduced, the higher-order method may be less accurate than Euler's method. Thus, in the error-stepsize control we drop to Euler's method when drastic stepsize reductions are required, as often occurs in the case of a corrector failure. This also suggests that local extrapolation be abandoned and that the order not be raised when the stepsize is being decreased. The following pseudocode summarizes these computations.

Order and Stepsize Selection Algorithm. Given the step index k ($k = 0$ corresponds to the first step), the previous degree m_{k-1} and stepsize h_{k-1} (for $k \geq 1$), the maximum degree, $Maxm$, the current point w_k, the predictor information for computing $P_{k,i}$, for $0 \leq i \leq M := \min\{k, m_{k-1} + 2, Maxm + 1\}$ (see (2.6)–(2.8)), the following algorithm selects the degree m_k and the stepsize h_k for computing the predicted point $wp(k+1, m_k, h_k)$.

Input Variables and Parameters:

k:	index denoting that $(k+1)$ points $\{w_i\}_{i=0}^{k}$ on the path have been computed.
h_{k-1}:	the predictor stepsize from the previous step if $k \geq 1$.
w_k:	the current point on the curve.
$\phi_{k,i}$ $(i = 0, \cdots, M)$, $\psi_{k-1,i}$, and $\beta_{k-1,i}$ $(i = 0, \cdots, m_{k-1}+1)$:	predictor information from previous steps.
$fail$:	a logical variable indicating corrector failure at stepsize $hfail$.
$hfail$:	attempted stepsize if $fail$ = true.
m_{k-1}:	the predictor degree selected on the previous step without local extrapolation.
Erel, Eabs:	relative and absolute predictor error tolerances, as appearing in the step control formula (4.1), e.g., Erel = Eabs = 0.01.
$Maxm$:	the maximum degree allowed for the predictor. This corresponds to a $Maxm+1$ step predictor which relies on a $Maxm+2$ step predictor for error control and local extrapolation, e.g., $Maxm := 4$.
$hmin$, $hmax$:	minimum and maximum steplengths allowed, e.g., $hmin = 0.000001$ and $hmax = 100.0$.

Output Variables:

h_k:	predictor stepsize selected.
m_k:	predictor order selected.

Algorithm:

1. If $k = 0$, do 1a-1c:
   1a. $h_0 = (\sqrt{\text{Eabs}} + \|w_0\|_2\sqrt{\text{Erel}})/2$
   1b. $m_0 = 0$
   1c. Return
2. If $k \geq 1$ do 2a-2d:
   2a. If fail = true, then go to 2c.
   2b. Select the predictor order using (4.1) and comparing consecutive orders:
      2b.1. Set $m := m_{k-1}$.
      2b.2. If $m < Maxm$ and $k \geq 2$, then compute γ_{m+1} and γ_m, and set
      $m_k := \text{argmax}\{\gamma_{m+1}, \gamma_m\}$
      $\{m_k$ is either m or $m+1\}$.
      2b.3. If $\gamma_{m_k} < 1$, then $m_k := m$ { Do not raise order if stepsize is decreased }.
      2b.4. If $m \geq 1$, then compute γ_{m_k} and γ_{m-1}, and set
      $m_k := \text{argmax}\{\gamma_{m_k}, \gamma_{m-1}\}$
      { Choose between orders m_k and $m-1$}.
      2b.5. If $m = Maxm \geq 2$, then compute γ_{m-2} and γ_{m_k}, and set
      $m_k := \text{argmax}\{\gamma_{m-2}, \gamma_{m_k}\}$
      { Choose between orders m_k and $m-2$}.
   2c. Select stepsize using (4.1) to compute γ_{m_k}:
      2c.1. $h_k := \text{Min}\{\text{Max}\{0.1, \gamma_{m_k}\}, 10.0\}h_{k-1}$.
      2c.2. If $fail$ = true, then $h_k := \min\{hfail, h_k\}$
      { If corrector failed, do not increase h}.
      2c.3. $h_k := \min\{hmax, \max\{h_k, hmin\}\}$ { Limit the stepsize }.
   2d. Return

The current choice of the initial stepsize h_0 is a simple and conservative one, and we would conjecture that more sophisticated starting schemes, such as those presented by Watts [46]–[48] for ordinary differential equation solvers, would yield considerable improvement. No doubt, predictor-corrector continuation codes could, in general, benefit from improved starting schemes. The issues, however, are somewhat different from those in ordinary differential equations. The main requirement for the starting stepsize is that the first step should yield a predicted point within the domain of convergence of the Newton-like method used in the correction process. If this requirement is not satisfied, then the stepsize is reduced until it is, resulting in many Newton-like corrections and considerable inefficiency. In the current methodology, an initial step which is a few orders of magnitude too small is adjusted within a few steps since we allow a large (10 times) change in the stepsize at each step. The issues are thus clouded by the fact that each step is equivalent to a function evaluation and most robust starting schemes for ordinary differential equations require several function evaluations (Watts's scheme requires four). Nevertheless, this is a topic that needs further investigation.

5. The corrector and the computation of the tangent. Having completed the discussion of the prediction phase and error-stepsize control, we now turn to a discussion of the corrector phase and the computation of the tangent to the curve. The corrected point on the curve is the solution of the augmented system of equations (3.1) and (3.2), i.e., $F(w) = 0$ and $N_1(w) = 0$, where $F : \mathbb{R}^{n+1} \to \mathbb{R}^n$ and $N_1 : \mathbb{R}^{n+1} \to \mathbb{R}$. The predicted point is usually used as an initial approximation to a solution of these equations. Methods ranging from Newton, Newton-like, or quasi-Newton methods (which may or may not employ a trust region or line search as a globalizer) to conjugate gradient methods may be used to solve this system of nonlinear equations. Some of

these methods are discussed in the reviews of Watson [43] and Allgower and Georg [1] as well as the work of Rheinboldt [33] and Keller [16]–[19]. Also, specific linear algebra methods have been developed to take advantage of the special structure of this system, but it is not our intent to defend or recommend one class of methods over another. For completeness, we describe the method used here, which is the bordering algorithm of Keller [18]. This method preserves the underlying matrix structure that may be present in problems with special structure, but theoretically it can fail at or near a singularity. In practice it performs reliably without modification; however, many additional refinements to this general procedure have been suggested in the literature [7] to overcome these deficiencies. We have also experimented with damped Newton's method and the use of the augmenting equation (3.3). Due to the philosophy of closely following the curve, we have not found these latter two procedures to give any significant improvement. For example, we have yet to find a problem for which damped Newton's method improves efficiency or reduces the number of corrector failures. The reason is that in the current methodology failures are almost always associated with the lack of a solution to (3.1), which most often occurs near sharp turns in the path.

The computation of the tangent to the curve is based on the fact that the tangent lies in the nullspace of the Jacobian matrix $D_w F(w)$, which we assume to be of full rank. This tangent is determined only to within a multiplicative constant, and thus the scaling and orientation of this vector must be specified to remove this nonuniqueness. Again, there are several methods for deciding on magnitude and orientation as discussed in the review of Allgower and Georg [1], as well as the work of Watson [42], Keller [16], and Rheinboldt [33]. Although any of these methods can be used with the current predictors, we have used Keller's bordering algorithm [16]–[19], which is briefly described next.

If $w = \binom{u}{\lambda}$, where λ is the parameter in the problem and $d_{k,m}(h_k) = \binom{\theta_k}{\rho_k}$, then (3.1) can be written as

$$\begin{bmatrix} F(w) \\ N(w) \end{bmatrix} = \begin{bmatrix} F(u,\lambda) \\ (u - up_{k+1})^T\theta_k + (\lambda - \lambda p_{k+1})\rho_k \end{bmatrix} = 0. \tag{5.1}$$

Furthermore, the tangent T satisfies

$$[D_u F(u_{k+1},\lambda_{k+1}) : D_\lambda F(u_{k+1},\lambda_{k+1})]T(u_{k+1},\lambda_{k+1}) = 0, \tag{5.2}$$

where (u_{k+1},λ_{k+1}) is the corrected point on the curve. Before explaining Keller's bordering algorithm, it is insightful to first recall Newton's method for the system (5.1) wherein we start from an initial approximation

$$\begin{pmatrix} u^0 \\ \lambda^0 \end{pmatrix} = \begin{pmatrix} up_{k+1} \\ \lambda p_{k+1} \end{pmatrix} = wp_{k+1}$$

and generate a sequence of points $\left\{\binom{u^i}{\lambda^i}\right\}$ where $\binom{u^{i+1}}{\lambda^{i+1}}$ is obtained from $\binom{u^i}{\lambda^i}$ by first solving the system

$$\begin{bmatrix} D_u F & D_\lambda F \\ \theta_k^T & \rho_k \end{bmatrix} \begin{bmatrix} \triangle u^i \\ \triangle \lambda^i \end{bmatrix} = - \begin{bmatrix} F(u^i,\lambda^i) \\ N(u^i,\lambda^i) \end{bmatrix} \tag{5.3a}$$

and then updating

$$\begin{pmatrix} u^{i+1} \\ \lambda^{i+1} \end{pmatrix} = \begin{pmatrix} u^i \\ \lambda^i \end{pmatrix} + \begin{pmatrix} \triangle u^i \\ \triangle \lambda^i \end{pmatrix}. \tag{5.3b}$$

On each iteration the matrix D_uF and vector $D_\lambda F$ are evaluated at the point $\binom{u^i}{\lambda^i}$ for Newton's method but are left fixed at $\binom{u^0}{\lambda^0}$ for the chord method. In Keller's bordering algorithm, one first solves for z^i and v^i from the systems

$$[D_uF]z^i = -F(u^i, \lambda^i), \tag{5.4a}$$

$$[D_uF]v^i = -D_\lambda F, \tag{5.4b}$$

respectively. Then any solution to (5.3a) has the form

$$\triangle u^i = z^i + \triangle\lambda^i v^i \tag{5.4c}$$

where

$$\triangle\lambda^i = -(N(u^i, \lambda^i) + \theta_k^T v^i)/(\theta_k^T z^i + \rho_k). \tag{5.4d}$$

The iterate can then be updated according to (5.3b). Note that (5.4a–d) enable us to obtain $\binom{\triangle u^i}{\triangle\lambda^i}$ while exploiting any special structure in D_uF. For this reason (5.4a–d) can be less expensive than the direct factorization of the matrix in (5.3a).

Another advantage of the bordering algorithm is that the tangent to the path is easily obtained at little extra expense. Suppose

$$\begin{pmatrix} u^{i+1} \\ \lambda^{i+1} \end{pmatrix}$$

is accepted as the corrected point on the curve so that

$$w_{k+1} = \begin{pmatrix} u_{k+1} \\ \lambda_{k+1} \end{pmatrix} = \begin{pmatrix} u^{i+1} \\ \lambda^{i+1} \end{pmatrix},$$

and the factorization of D_uF is still available from the previous step. Then a good approximation to the unit vector T which solves $[D_uF : D_\lambda F]T = 0$ is

$$T = \pm \begin{pmatrix} v^i \\ 1 \end{pmatrix} \Big/ \sqrt{1 + \|v^i\|_2^2}. \tag{5.5}$$

The sign determines the orientation of the continuation and is subject to change only when $\det(D_uF)$ changes sign. If this determinant changes sign, but that of the Jacobian of the augmented system (5.1) does not, then we declare that a fold point singularity has been passed and change the orientation of T. If both determinants change sign, then a bifurcation point is declared [16]. (These determinants can be computed cheaply from the factorization of D_uF if the elimination and pivoting steps are also performed on $D_\lambda F$. A simple procedure used by Watson [44] to maintain orientation is based on maintaining a positive $T_k^T T_{k+1}$ and this works well when no bifurcation points are present.) Next, if w_{k+1} is a fold point singularity, then D_uF is singular and $D_\lambda F$ is not in the range of D_uF, so that v^i must be defined alternately as a null vector of D_uF, and each "1" in formula (5.5) must be changed to "0." Keller [18] has shown, however, that (5.4a–d) and (5.5) remain effective computationally even when D_uF is ill-conditioned as long as the pivoting strategy used gives a small last pivot. In this case, (5.4b) is essentially a step of inverse iteration for a null vector, and relative to the size of the computed v^i, the 1's in (5.5) become negligible.

Since the factorization of the matrix $D_u F$ is the major expense in many continuation problems, the simplified Newton iteration in which the Jacobian is evaluated and factored only at the predicted point (the chord method) gives increased efficiency [16], [25], [33], [36]. In this procedure one exchanges the quadratic rate for a linear rate of convergence, but retains the same Kantorovich domain of convergence [27]. However, the contraction factor in the linear rate of convergence decreases with increased accuracy in the predictor.

When using the above Newton iteration, a termination criterion must be provided to determine when an iterate w_{k+1}^i is to be accepted as the corrected approximation to $w_{k+1} = w(s_{k+1})$ or when to declare that the iteration is to be stopped due to the lack of convergence. Our algorithm employs a user-specified error corrector tolerance, Ecor, and maximum iteration limit, $Maxit$ (e.g., $Maxit = 7$). The corrector tolerance is usually chosen such that $[(\text{Eabs} + \text{Erel})/2]^2 \leq \text{Ecor}$, and Ecor > Macheps, where Macheps denotes machine precision or unit roundoff. The idea is to require roughly the error reduction of one iteration of a quadratically convergent process, so that the minimal Ecor will require one or two iterations of the corrector, assuming the predictor has satisfied its given tolerance. For our termination criterion we also demand that corrector iterate w^{i+1} satisfy

$$\text{(5.6a)} \quad \|F(w^{i+1})\|_\infty \leq \text{Ecor} \cdot \max\{\|F(w^0)\|, 1\},$$

$$\text{(5.6b)} \quad |[w^{i+1} - w^i](j)| \leq \text{Ecor} \cdot \max\{|typw|, |[w^{i+1}](j)|\} \quad \text{for } j = 1, \cdots, n+1,$$

where $w^{(0)} = wp_{k+1}$ is the predicted point and $typw$ is a pre-chosen "typical value" of $[w^{i+1}](j)$, which is usually set to 1.0 [10]. This requires (roughly) that the change in each component of w^i relative to w^{i+1} be less than Ecor. If these criteria are not satisfied for some $i + 1 \leq Maxit$, the iteration is aborted and termed a corrector failure. We also stop the iteration and declare a corrector failure if either

$$\text{(5.6c)} \qquad \|w^{i+1} - w^i\|_\infty \geq \epsilon_1 \|w^i - w^{i-1}\|_\infty, \qquad i \geq 0$$

or

$$\text{(5.6d)} \qquad \|F(w^{i+1})\|_\infty \geq \epsilon_2 \|F(w^i)\|_\infty, \qquad i \geq 0,$$

where ϵ_1 and ϵ_2 are typically chosen to be 1.5. As can happen with any stepsize control, the predicted point may not lie within the domain of attraction of Newton's method for (3.1) and thus a method for dealing with a corrector failure is needed.

In case of a failed corrector we immediately drop to Euler's method and recompute the stepsize h_k using the error-stepsize control, with the additional constraint that the new stepsize not be greater than the failed stepsize. Then we use an Euler predictor with this stepsize and try the corrector again from this new point. In case of subsequent failures or if the first failure involved an Euler predictor, we continue halving the stepsize (thus circumventing the error control procedure) until corrector convergence is obtained or some minimum stepsize $h_{\min}$ is reached, in which case the entire algorithm is stopped. Another modification to the stepsize control occurs when we want to compute a special point on the curve. For higher-order predictors we must solve a nonlinear equation to find the stepsize $\bar{h}$ for which the predicted point has the same parameter value as the special point; however, Newton's method can be used effectively to find this $\bar{h}$. Then we perform a vertical correction that involves the use of the augmenting equation (3.4). A potentially more robust procedure using inverse interpolation can be found in the work of Watson, Billups, and Morgan [44].

6. A high level pseudocode. We summarize the discussion of the previous three sections by providing in this section an outline of a continuation algorithm which incorporates our predictor strategy.

It should be emphasized that many different linear algebra methods may be used in the corrector and the tangent computation, but we have used those described in the previous section to obtain the numerical results reported for our code ABCON in §7.

Algorithm: Predictor-Corrector Continuation Using Variable Order Adams–Bashforth Predictors. Given $F : \mathbb{R}^{n+1} \to \mathbb{R}^n, w_0 = \binom{u_0}{\lambda_0}$ such that $F(w_0) = 0$ and parameter bounds λ_L and λ_R with $\lambda_0 \in (\lambda_L, \lambda_R)$, the following algorithm obtains a computational trace $\{w_k\}_{i=0}^{K}$ of a segment of the solution path of the equation $F(w) = 0$, terminating with $[w_K]_{n+1} = \lambda_R$ (or λ_L) or $K = Kmax$.

Input Parameters (with default values):

Orient:	determines the orientation of the path and has a value of ± 1 as appears in (5.5).
λ_L, λ_R :	lower and upper bounds on the continuation parameter λ.
$Kmax$:	the maximum number of points computed (5000).
Erel, Eabs:	relative and absolute predictor error tolerances, as appearing the step control formula (4.1) (0.01,0.01).
Ecor:	error tolerance for the computed points, as appears in the acceptance criterion (5.6a) and (5.6b) (.0001).
Efin:	error tolerance for final point (at $\lambda = \lambda_L$ or λ_R); Ecor is reset to this value for the final correction (0.0001)
w_0:	the initial point on the path satisfying $\lambda_0 \in (\lambda_L, \lambda_R)$ and $\|F(w_0)\| \leq$Ecor.
$Maxm$:	the maximum degree allowed for the predictor. This corresponds to a $Maxm + 1$ step predictor which relies on a $Maxm + 2$ step predictor for error control (4).
$hmin$, $hmax$:	minimum and maximum steplengths allowed (0.000001,100.0).
$Maxit$:	maximum number of Newton-like iterations in one correction attempt (7).

Output Parameters:

$\{w_k\}_{k=1}^{K}$:	Computed points on the path with $K = Kmax$ or $[w_K]_{n+1} = \lambda_L$ or λ_R.
$\{\triangle s_k\}_{k=0}^{K-1}$:	Computed arclength stepsizes between w_k and w_{k+1}.
$\{T_k\}_{k=0}^{K-1}$:	Computed tangents to the path at w_k.

Algorithm:

1. Set $fail$:=false and EC := Ecor.
2. Compute tangent $T_0 = T(w_0)$ via (5.4b) and (5.5) (or other methods) with the sign of Orient; set $\varphi_{0,0} := T_0$.
3. Compute the determinant of $S_0 = D_u F(w_0)$ and of the matrix

$$L_0 = \begin{bmatrix} D_w F(w_0) \\ D_w N(w_0) \end{bmatrix}$$

   using the factorization of $D_u F(w_0)$ from step 2.
4. Set $k = 0$
5. While $k \leq Kmax$ do steps 5–18

{ Prediction phase occurs in steps 6–9 }

6. If $k \geq 1$, then update predictor information:
   6a. Reset $h_{k-1} := \triangle s_{k-1}$.
   6b. Compute the $\varphi_{k,i}$ for forming $P_{k,i}(s)$ for $i = 0, \cdots, \text{Min}\{m_{k-1} + 2, k, maxm + 1\}$, using T_k, $\triangle s_{k-1}$, the $\varphi_{k-1,i}$, and the formulas in §2.
7. Select the predictor order m_k and stepsize h_k using the algorithm in §4.
8. If $k \geq 1$ and $m_k \geq m_{k-1}$, then $m := m_k + 1$ { use local extrapolation }, else $m := m_k$.
9. Compute the predictor:
   9a. Compute the predictor using (2.5)–(2.7) and (2.9):
      9a.1. $wp := wp(k + 1, h_k, m)$.
      9a.2. $d := d_{k,m}(h_k)$.
   9b. If $\lambda p \ (= [wp]_{n+1}) \leq \lambda_L$ (respectively, $\lambda p \geq \lambda_R$), then compute target point predictor:
      9b.1. Compute $\bar{h}$ with $\lambda p = [wp(k + 1, \bar{h}, m)]_{n+1} = \lambda_L$ (respectively, λ_R) using Newton's method.
      9b.2. $h_k := \bar{h}$.
      9b.3. $wp := wp(k + 1, h_k, m)$ (using 2.5–2.7,2.9).
      9b.4. $d := e_{n+1}$ (direction for correction with λ fixed).
      9b.5. $m := 0$, EC := Efin.

{ Correction, tangent, and arclength computation occur in steps 10–17 }

10. Perform and monitor chord, Newton, or quasi-Newton iterations on system (5.1).
    10a. Set $i := 0$ and $w^0 := wp$
    10b. While $i \leq Maxit$ Do 10b.1–10b.5:
       10b.1. Compute z^i, v^i, and $\triangle w^i := \binom{\triangle u^i}{\triangle \lambda^i}$ using equations (5.4a–d)
       10b.2. Update $w^{i+1} := w^i + \triangle w^i$
       10b.3. If w^{i+1} satisfies (5.6a) and (5.6b), set $w_{k+1} := w^{i+1}$ and Go To 12.
       10b.4. If w^{i+1} satisfies (5.6c) (for $i \geq 1$) or (5.6d), declare a corrector failure and Go To 11.
       10b.5 $i := i + 1$ and Go To 10b.
11. Cut order and/or steplength and prepare for another corrector attempt.
    11a. $Fail$:=true, $hfail := h_k$, EC := Ecor.
    11b. If $m_k > 0$, then reset $m_k := 0$ and Go To 7.
    11c. $h := h/2$.
    11d. If $2h \leq hmin$, then STOP since the run failed.
    11e. Go To 9.
12. If $\lambda_{n+1} = [w_{k+1}]_{n+1} = \lambda_R$ (or λ_L), then target has been reached, OUTPUT (w_{k+1}), and STOP.
13. Compute tangent T_{k+1} via (5.5) using the v^i from the above corrector iteration, and with the sign of Orient.
14. Compute the determinant of $S_{k+1} = D_u F(w_{k+1})$ and of the matrix

$$L_{k+1} = \begin{bmatrix} D_w F(w_{k+1}) \\ D_w N(w_{k+1}) \end{bmatrix}$$

    from the last PLU factorization of $D_w F$ in step 9.
15. If $\det(S_k)\det(S_{k+1}) < 0$ and $\det(L_k)\det(L_{k+1}) > 0$, then declare that a fold point has been passed, set $T_{k+1} := -T_{k+1}$ and Orient := − Orient.
    If $\det(S_k)\det(S_{k+1}) < 0$ and $\det(L_k)\det(L_{k+1}) < 0$,

then declare that a bifurcation point has been passed.
16. Compute the corrected arclength $\triangle s_k$ by using h_k in the arclength correction algorithm in §3.
17. OUTPUT w_{k+1}, T_{k+1}, and $\triangle s_k$.
18. Set $k := k + 1$ and Go To 5.
19. STOP and OUTPUT ("Maximum Number of Steps Exceeded").

It should be observed that a Newton, quasi-Newton, or chord iteration could be used in step 10 above. Also, additional accuracy in the tangent may be obtained by solving (5.4b) with the derivatives evaluated at w_{k+1} and using the resulting solution in (5.5). Each of these entails additional computation.

7. Numerical examples. To examine the efficiency and robustness of the above methodology, we have developed a research code called ABCON for *A*dams–*B*ashforth *con*tinuation which employs two methods for solving the corrector equations (3.1) or (5.1). The first employs a full Newton method with tangents computed at the next-to-last corrected point and is called ABCON Full; the second uses a chord method with tangents computed at the predicted rather than corrected point and is called ABCON Chord, somewhat in analogy to Rheinboldt's PITCON Full and Chord [30], [31], [33]. ABCON was "tuned" by numerically computing the penalty paths for constrained optimization [23], [28] and the path defined by the ellipse $au^2 + b\lambda^2 = 1$ for the ratio a/b ranging from 10^{-10} to 10^{10}. Eleven difficult small test problems were then chosen from the continuation literature [9], [36], [45], [49] to provide a basis for testing the robustness and efficiency of ABCON as well as a numerical comparison with the state-of-the-art software packages HOMPACK by Watson, Billups, and Morgan [44] and PITCON by Rheinboldt and Burkardt [30], [31]. (HOMPACK has three methodologies in the codes QF (augmented Jacobian), NF (normal flow), and DF (ordinary differential equations method) as well as sparse versions of these (QS, NS, and DS, respectively).) Since the test problems were small dense ones, the conclusions in this section are necessarily confined to this class of problems; however, to give some indication of potential performances for other problem classes, we report the number of computed points on the curve, the number of matrix factorizations, the number of function evaluations, the number of linear systems solved, the arclength of the curve, tolerances used, and the number of runs required to successfully solve a problem. Before continuing with a discussion of the test problems and the numerical results, we first discuss the design objectives in these three codes and some of the implications.

A philosophy behind the use of these Adams–Bashforth predictors and with the developed error-stepsize control has been to follow the curve "closely" for robustness and to take shorter steps, so that only two or three corrections back to the curve are generally required. The second correction is essentially used only to check the accuracy of the first full Newton correction, and thus the power of Newton's method in the correction process is not utilized. ABCON is a research code tuned on small problems, while HOMPACK and PITCON are production software packages intended and tuned for the solution of a wide variety of problems. HOMPACK focuses on nonlinear equations via homotopy methods, particularly the regularizing homotopy, and PITCON, on large sparse problems. Since the design objectives of these three codes differ, we should not make any general conclusions about their relative efficiency and robustness; the test set is much too small for such a comparison. Instead, the objective is to investigate the viability of the use of the Adams–Bashforth predictors with the developed error-stepsize control in continuation codes.

As stated earlier the focus in this work has been the development of Adams–

Bashforth predictors and a corresponding error-stepsize control. Thus we have not tried to develop any of the special linear algebra methods [43] for different problem classes. The linear algebra methods in the development of ABCON were based on the bordering algorithm of Keller [17], which is generally applicable, and Gaussian elimination with scaled partial pivoting to solve the reduced problem in this bordering algorithm. The test problems reflect this development in that they are restricted to small dense ones with dimensions ranging from 2–50; however, they are difficult ones used in the continuation literature [9], [36], [44], [45] to test continuation methodologies. Ten of the 11 problems are homotopy problems; eight of these were taken from the papers of Watson, Billups, and Morgan [44], [45] and were used in testing HOMPACK. With regard to the remaining three test problems, number 11 in the following list was taken from the paper of Heijer and Rheinboldt [9] in which the error-stepsize control for PITCON was developed. Problems 3 and 4 can be found in the work of Schwetlick and Cleve [36] with the former being a parameter continuation problem and the latter a homotopy problem. The 11 test problems are as follows:

1. The Watson curve with $n = 10$ [45]: $F_i(u,\lambda) = u_i - \lambda \exp[\cos(i \cdot \Sigma_{k=1}^{n} u_k)]$ for $i = 1, \cdots, n$ and $n = 10$. The path defined by $F = 0$ is traced from $\lambda = 0$ with $u = 0$ to $\lambda = 1$. Forty-six fold point singularities are encountered along the path.

2. The Watson curve with $n = 12$ [45]: This is the same as problem 1, except the number of equations is twelve (12) and 56 fold point singularities are encountered along the path.

3. A homotopy problem [36]. $F(u,\lambda) = f(u) - (1-\lambda)f(u_0)$, where $f(u) = [D_u G(u)]^T G(u)$, $G(u)^T = (10(u_2 - u_1^2), 1 - u_1, 3\sqrt{10}\ (u_4 - u_3^2), 1 - u_3, \sqrt{10}\ (u_2 + u_4 - 2), (u_2 - u_4)/\sqrt{10}\ )$, and $u_0^T = (-3,-1,-3,-1)$. The solution path is traced from $(u_0^T, 0)$ to $(1,1,1,1)^T$. Four fold points were detected for λ between 0.999 and 1.0.

4. $F_1(u,\lambda) = (u_1 - u_3)/10^4 + (u_1 - u_2)/39 + (u_1 + \lambda)/51$
   $F_2(u,\lambda) = (u_2 - u_6)/10 + (u_2 - u_1)/39 + I(u_2)$
   $F_3(u,\lambda) = (u_3 - u_1)/10^4 + (u_3 - u_4)/25.5$
   $F_4(u,\lambda) = (u_4 - u_3)/25.5 + u_4/0.62 + u_4 - u_5$
   $F_5(u,\lambda) = (u_5 - u_6)/13 + u_5 - u_4 + I(u_5)$
   $F_6(u,\lambda) = (u_6 - u_2)/10 + (u_6 - u_5)/13 + u_6 - U(u_3 - u_1)/0.201$
   $I(x) = 5.6 \cdot 10^{-8}[\exp(25x) - 1]$, $U(x) = 7.65 \arctan(1962x)$. The path defined by $F = 0$ is traced from $\lambda = 0$ to the first point with $\lambda = 1$. Two fold points were encountered [36].

5. $F(u,\lambda) = u - \lambda f(u)$ where $f_i(u) = (\Sigma u_k^3 + i)/2n$ for $i = 1, \cdots, n$ and $n = 10$. The path is traced from $\lambda = 0$ with $u = 0$ to $\lambda = 1$ [45].

6. $F(u,\lambda) = u - \lambda f(u)$ where $f_1(u) = 0.01(u_1 + u_2 + 1)^3$, $f_i(u) = 0.01(u_{i-1} + u_i + u_{i+1} + 1)^3$ for $i = 2, \cdots, n-1$, $f_n(u) = 0.01(u_{n-1} + u_n + 1)^3$, and $n = 10$. The path is traced from $\lambda = 0$ with $u = 0$ to $\lambda = 1$ [45].

7. $F(u,\lambda) = u - \lambda f(u)$ where $f_1(u) = u_1 - \prod_{j=1}^{n} u_j + 1$ and $f_i(u) = -\Sigma_{j=1}^{n} u_j + n + 1$ for $i = 2, \cdots, n$, and $n = 10$. The path is traced from $\lambda = 0$ with $u = 0$ to $\lambda = 1$ [45].

8. Same as problem 7 except that the dimension $n = 25$.

9. Same as problem 8 except that the dimension $n = 50$.

10. The Freudenstein–Roth problem [9], [49] with a regularizing homotopy:

$$f(u_1, u_2) = \begin{bmatrix} u_1 + 5u_2^2 - u_2^3 - 2u_2 - 13 \\ u_1 + u_2^2 + u_2^3 - 14u_2 - 29 \end{bmatrix} = 0,$$

is solved using the regularizing homotopy $F(u, \lambda) = \lambda f(u) + (1 - \lambda)(u - u_0)$ with $u_0 = (15, -2)^T$. The solution is $u^* = (5, 4)^T$ and two fold points are passed.

11. The Freudenstein–Roth problem [9], [49] with a Newton global homotopy:

$$f(u_1, u_2) = \begin{bmatrix} u_1 + 5u_2^2 - u_2^3 - 2u_2 - 13 \\ u_1 + u_2^2 + u_2^3 - 14u_2 - 29 \end{bmatrix} = 0,$$

is solved using the Newton global homotopy $F(u, \lambda) = f(u) - (1 - \lambda)f(u_0)$ and with $u_0 = (15, -2)^T$. In each case the solution $u^* = (5, 4)^T$ is computed and two fold points are passed.

The next objective is to describe the test conditions and strategies. All computations were performed on a CDC Cyber 180/840 computer in single precision which has a unit roundoff (machine epsilon) of approximately 7×10^{-15}. The minimum and maximum stepsize limits in all codes was set to $hmin = 10^{-7}$ and $hmax = 100$, respectively. All problems were attempted with the error tolerance for the computed solution set to 10^{-4} for each code (abserr = relerr = 10^{-4} for PITCON, arcae = arcre = ansre = 10^{-4} for HOMPACK, and Ecor = Efin = 10^{-4} for ABCON). For ABCON, predictor error tolerances were set to Eabs = Erel = 10^{-2}, the maximum number of corrector iterations to 7, and the maximum order, $Maxm$, used in the Adams–Bashforth predictors was set to 4. If a run at this error tolerance was unsuccessful, the error tolerance was divided by 10 and the problem was rerun. This was repeated until a successful run was obtained or an error tolerance of 10^{-12} led to failure. In the case of ABCON, which is the only code that allows explicit control of the error in the predictors, the predictor error tolerances were also divided by 10 after each failure. A run was considered successful if the correct final point was computed to at least three significant digits, no error flag was returned by the code, arclength was reported with less than five percent error (PITCON does not report arclength), and the proper number of fold (limit, turning) points were detected (HOMPACK does not report fold points). PITCON and HOMPACK were not penalized for numbers which they did not report and the extra work required by PITCON to compute the number of fold points was not counted in the tables below. ABCON was, however, penalized for not reporting each of these correctly and the extra work required to compute the number of fold points is included in the reported numbers. Reliable and accurate values of these quantities for each problem were obtained by running all codes on the problems using smaller and smaller error tolerances.

Since the test problems are small dense ones ranging in dimension from 2–50, several different performance measurements are presented in the table below to give some indication of the potential performance in other problem classes. These include the number of computed points (PTS) on the curve, the number of Jacobian evaluations and factorizations (NJF), the number of linear systems solved using these factorizations (NLS), the number of function evaluations (NFE), and the computed arclength (ARC) for each run. Notes are also given which indicate the tolerances used, the number of runs required to obtain success, and the reason for failure. The average counts are summarized at the end of the table; however, the homotopy problem 11 is omitted from these averages for HOMPACK, since it is not designed to solve problems where $\lambda < 0$ on a segment of the homotopy path.

TABLE
Numerical comparisons.

CODE	*PTS*	*NJF*	*NFE*	*NLS*	*ARC*	*TOL*	*#Runs*	*Notes*
Example 1 (n=10)								
ABCON Full	572	1280	1869	1869	87.38	10^{-5}	2	d
ABCON Chord	354	385	1406	1406	87.26	10^{-4}	1	
PITCON Full	282	1478	1534	1478	*	10^{-10}	7	d
PITCON Chord	218	541	1581	1475	*	10^{-5}	2	d
HOMPACK QF	333	417	3133	2672	87.05	10^{-6}	3	b, g
HOMPACK NF	258	1062	1062	2124	85.9	10^{-10}	7	b, g
HOMPACK DF	2603	5307	5311	5307	87.5	10^{-8}	5	b, g
Example 2 (n=12)								
ABCON Full	716	1593	2335	2335	108.47	10^{-5}	2	d
ABCON Chord	438	473	1696	1696	108.3	10^{-4}	1	
PITCON Full	254	1272	1387	1272	*	10^{-5}	2	c, d, g
PITCON Chord	280	677	2625	2508	*	10^{-8}	5	c, d, g
HOMPACK QF	902	949	7206	6217	108.10	10^{-9}	6	b, g
HOMPACK NF	323	1275	1275	2550	108.73	10^{-9}	6	b, g
HOMPACK DF	3172	6466	6471	6466	108.21	10^{-8}	5	b, g
Example 3 (n=4)								
ABCON Full	168	342	509	509	16.73	10^{-7}	4	d
ABCON Chord	193	195	593	593	16.73	10^{-7}	4	d
PITCON Full	FAILED AT ALL TOLERANCES						9	c, d, g
PITCON Chord	39	96	303	205	*		1	
HOMPACK QF	120	124	851	851	16.72	10^{-11}	8	d, e, g
HOMPACK NF	52	236	236	472	16.66	10^{-11}	8	d, e, g
HOMPACK DF	173	361	361	361	16.73	10^{-5}	2	b, g
Example 4 (n=6)								
ABCON Full	41	104	148	148	51.69	10^{-5}	2	d
ABCON Chord	55	60	224	224	51.71	10^{-5}	2	d
PITCON Full	16	76	80	76	*	10^{-4}	1	
PITCON Chord	28	67	305	293	*	10^{-4}	1	
HOMPACK QF	61	65	284	284	51.67	10^{-8}	5	b, f
HOMPACK NF	64	91	91	182	51.67	10^{-4}	1	
HOMPACK DF	95	203	203	203	51.83	10^{-4}	1	
Example 5 (n=10)								
ABCON Full	3	7	9	9	1.41	10^{-4}	1	
ABCON Chord	3	4	9	9	1.41	10^{-4}	1	
PITCON Full	3	11	12	11	*	10^{-4}	1	
PITCON Chord	3	6	12	11	*	10^{-4}	1	
HOMPACK QF	3	4	16	12	1.45	10^{-4}	1	
HOMPACK NF	4	12	12	24	1.45	10^{-4}	1	
HOMPACK DF	13	30	31	30	1.45	10^{-4}	1	
Example 6 (n=10)								
ABCON Full	3	6	8	8	1.0005	10^{-4}	1	
ABCON Chord	3	4	8	8	1.0005	10^{-4}	1	
PITCON Full	3	9	10	9	*	10^{-4}	1	
PITCON Chord	3	7	10	9	*	10^{-4}	1	
HOMPACK QF	3	4	14	10	1.001	10^{-4}	1	
HOMPACK NF	6	13	13	26	1.001	10^{-4}	1	
HOMPACK DF	12	28	29	28	1.001	10^{-4}	1	
Example 7 (n=10)								
ABCON Full	7	20	26	26	3.71	10^{-4}	1	
ABCON Chord	7	8	26	26	3.71	10^{-4}	1	
PITCON Full	11	50	55	50	*	10^{-4}	1	
PITCON Chord	6	15	54	51	*	10^{-4}	1	

TABLE (*cont.*)

HOMPACK QF	7	8	58	50	3.71	10^{-5}	2	c, g
HOMPACK NF	7	25	25	50	3.67	10^{-4}	1	
HOMPACK DF	26	56	57	56	3.72	10^{-4}	1	
Example 8 (n=25)								
ABCON Full	10	35	46	46	5.71	10^{-4}	1	
ABCON Chord	10	12	51	51	5.67	10^{-4}	1	
PITCON Full	15	65	70	65	*	10^{-4}	1	
PITCON Chord	12	29	67	62	*	10^{-4}	1	
HOMPACK QF	10	11	79	68	5.67	10^{-6}	3	c, g
HOMPACK NF	10	26	26	52	5.66	10^{-4}	1	
HOMPACK DF	27	61	62	61	5.69	10^{-4}	1	
Example 9 (n=50)								
ABCON Full	11	32	44	44	7.86	10^{-4}	1	
ABCON Chord	11	14	51	51	7.82	10^{-4}	1	
PITCON Full	17	71	77	71	*	10^{-4}	1	
PITCON Chord	19	46	88	80	*	10^{-4}	1	
HOMPACK QF	11	13	67	53	7.81	10^{-4}	1	
HOMPACK NF	12	27	27	54	7.76	10^{-4}	1	
HOMPACK DF	32	73	74	73	7.86	10^{-4}	1	
Example 10 (n=2)								
ABCON Full	33	95	130	130	33.94	10^{-4}	1	
ABCON Chord	44	50	186	186	32.75	10^{-4}	1	
PITCON Full	17	95	100	95	*	10^{-8}	5	d
PITCON Chord	18	43	116	109	*	10^{-4}	1	
HOMPACK QF	42	50	249	194	32.69	10^{-4}	1	
HOMPACK NF	43	78	78	156	32.67	10^{-4}	1	
HOMPACK DF	202	420	423	420	32.75	10^{-6}	3	b, g
Example 11 (n=2)								
ABCON Full	51	126	182	182	105.39	10^{-4}	1	
ABCON Chord	50	58	203	203	104.95	10^{-4}	1	
PITCON Full	13	60	64	60	*	10^{-6}	3	d
PITCON Chord	14	33	131	126	*	10^{-10}	7	c, g
HOMPACK QF								b, g, h
HOMPACK NF								b, g, h
HOMPACK DF								b, g, h

	Average counts				
	PTS	NJF	NFE	NLS	RUNS
ABCON Full	146.8	330.9	482.4	482.4	1.5
ABCON Chord	106.2	114.8	404.8	404.8	1.4
PITCON Full	63.1	318.7	338.9	318.7	2.9
PITCON Chord	58.2	141.8	481.1	448.1	2.0
HOMPACK QF	149.2	164.5	1195.7	1041.1	2.7
HOMPACK NF	77.9	284.5	284.5	569	2.8
HOMPACK DF	635.5	1300.5	1302.2	1300.5	2.1

* - results not reported by this code.
a - over five percent error in the computed arclength.
b - lost path, did not track all the way to $\lambda = 1$.
c - failed in final point correction.
d - did not detect proper number of limit points.
e - final solution wrong.
f - execution terminated due to overflow or an infinite result.
g - code stopped and returned error flag.
h - code is not intended for this type of problem where $\lambda < 0$ on a segment of the homotopy path.

The performance measurements presented in this table illustrate the overall efficiency of these Adams–Bashforth predictors. First, the average number of runs

required to "solve" the problem to the given tolerances may in itself be a most significant number. ABCON Chord was the most efficient with an average number of runs per problem of 1.4 followed by ABCON Full (1.5), but with PITCON Chord (2.0) and HOMPACK DF (2.1) close behind. Of the two problems on which additional runs of ABCON Chord were required, the answer was found correctly on the first run, but the code detected too many fold points. For example, in problem 3 the solution was computed efficiently with NJF = 62 and NLS = 219 on the first run as opposed to the final count of NJF = 195 and NLS = 593 on the fourth run. The version of ABCON that uses a *full* Newton method appears to be less efficient and no more robust than the *chord* version. Note that PITCON also shows this phenomenon with the difference in failure rate being even more pronounced. In ABCON the predicted point is close to the curve, resulting in only two or three corrections per step, but the second correction is generally used only to verify the accuracy of the first Newton correction in both the full and chord methods, so that the full power of Newton's method in the correction process is generally not utilized. The apparent lower efficiency of ABCON Full in comparison to ABCON Chord is also, curiously, due to larger stepsizes resulting from the more accurate tangents. These larger stepsizes caused the code to jump curves and to miss the fold points more often, resulting in additional reruns. If one considers only the examples on which ABCON Full and ABCON Chord used the same error tolerances, the full method does tend to compute fewer points and uses fewer solves. The tentative conclusion that we would like to draw from these numbers is that the Adams–Bashforth predictors, wherein the predictor error is controlled, may lead to a more robust and less sensitive method for continuation procedures.

Next, ABCON Chord appears to be more efficient in its use of Jacobian evaluations and factorizations (NJF) than the other codes. For problems in which NJF is a good measure of the efficiency, ABCON Chord is superior, using over 19 percent fewer NJF on problems 1–11 than the leading competitor, PITCON Chord. This conclusion is even more significant given that ABCON Chord computed nearly twice as many points and was more robust (1.4 versus 2.0 average runs required for success) than PITCON Chord. Thus in parameter studies of nonlinear systems wherein one uses matrix factorizations to detect singularities along the path, ABCON Chord is a method of choice. In terms of the number of function evaluations, the codes HOMPACK NF and PITCON Full required 33 percent and 13 percent fewer, respectively, but required at least twice as many runs as ABCON Chord to achieve a successful run.

We might argue that the number of linear systems solved is a good measure of the efficiency of these methods on larger-scale problems wherein one might use iterative methods. Before making this comparison, it is necessary to describe differences in matrix algebra in the various codes and to explain how we have counted NLS for each code. Both PITCON and HOMPACK QF solve an $(n+1) \times (n+1)$ system similar to (5.3) each time a corrector iterate or tangent is computed—the former by use of a PLU factorization and the latter by using a more stable but expensive QR factorization with column pivoting. In PITCON Chord and HOMPACK QF a factorization may be used in several back solves. HOMPACK NF and DF use a QR factorization on the $(n+1) \times (n)$ matrix $D_w F$ and back substitute using an $n \times n$ upper triangular block of R. To be consistent with the varying methods of computing tangents and/or iterates, we report in NLS the number of corrector iterates computed plus the number of tangents computed by each code, even though, when solving (5.3), ABCON uses the bordering algorithm which involves a PLU factorization of the $n \times n$ matrix $D_u F$ and which involves one or two back solves with this factorization. In the

case of ABCON Chord this gives the same number as the number of solves of (5.4a) plus the number of solves of (5.4b), but for ABCON Full, the two methods of counting give different results. Also, the cost of the quasi-Newton updates in HOMPACK QF, which for small to medium sized systems is comparable to the cost of a backsolve, is not counted.

For the number of linear systems solved, ABCON Chord is exceeded in efficiency by PITCON Full (21.3 percent fewer); however, this difference is slight in comparison to the large corresponding increase (278 percent) in the number of Jacobian factorizations (NJF) with double the failure rate (2.9 versus 1.4). If we restrict the comparison to those methods where a full Newton method is used in the correction process, then the winners are HOMPACK for NJF (14 percent fewer) and PITCON for NLS (33.9 percent fewer). (Such a full Newton method may be needed in the correction process for certain problem classes or specific applications.) Note that these comparisons of runs obtained at near optimal settings do not reflect the fact that the code PITCON Full (2.9) and HOMPACK NF (2.8) have nearly twice the failure rate as ABCON Full (1.5) and ABCON Chord (1.4). Thus the conclusions in this paragraph are somewhat clouded by the failure rates, by the fact that more experienced users of HOMPACK and PITCON may be able to solve a given problem with these packages without so many reruns, and the potential use of ABCON Chord even on large scale problems. (Recall that the second correction is generally used only to verify the accuracy of the first Newton correction in both ABCON Full and ABCON Chord.) However, the general trend of these failure rates support the obvious fact that following the solution path "closely" does in general produce a more robust code, which is especially important in parameter as opposed to homotopy continuation.

With regard to the number of points computed, ABCON Chord produced nearly double the number of points on the path than did the competing codes PITCON Full, PITCON Chord, and HOMPACK NF. While this may not be a direct advantage (additional points may always be computed by interpolation), it does provide an indirect advantage due to the added robustness obtained and the increased confidence we have in the details of the computed path. Although this may not be of interest in homotopy problems, the focus of HOMPACK, it is of fundamental interest in parametric studies of systems of nonlinear equations. Furthermore, we have found that the competing codes would more often jump over a portion of the curve, compute the wrong solution and arclength, or detect the wrong number of fold points without returning an error message. Thus following the path with an accurate predictor has its benefits, particularly with respect to natural parameter continuation.

Next, we note that Schwetlick and Cleve [36] report results for their code which uses Padé predictors on problems 1, 3, and 4, which give a slightly lower average NJF than those for ABCON; however, these results were not reproduced here since they were run at tolerances unknown to us and on a different machine. We were unable to determine from the published data whether the path was being properly traced by the criteria we imposed. Also, Ypma [49] has used a Hermite predictor and reported results on problem 11, but these are less efficient than those reported here for ABCON Chord.

Finally, we should comment on the vectorization of ABCON. A version of our code has been written to take advantage of the special vector hardware and FORTRAN 200 extensions resident on the CDC Cyber 205 supercomputer. A vectorized corrector which employs a PLU decomposition with scaled partial pivoting based on an extension of the ideas found in Dongarra, Gustavson, and Karp [13] was developed. The speedup obtained from these modifications of our original (compiler vectorized) code was over

40 times, and 99 percent of the speedup resulted from the explicit vectorization of the decomposition and solve routines (on problems with $n > 100$).

8. Summary and concluding remarks. Predictor-corrector methods for tracing solution paths of an underdetermined system of nonlinear equations $F(w) = 0$, where $F : \mathbb{R}^{n+1} \to \mathbb{R}^n$ have proved to be effective numerical procedures for various problems ranging from homotopy methods for nonlinear equations and fixed-point problems to parametric investigations of nonlinear equations arising in continuum mechanics, optimization, and dynamical systems. Robustness of these procedures is generally enhanced by following the solution path "closely," which can penalize efficiency. Although there has been much developed for these continuation algorithms, the use of Adams–Bashforth predictors has been recommended but not explored in a continuation procedure that employs a Newton-like correction back to the path. Thus variable order Adams–Bashforth predictors with an adaptive error-stepsize control for an arclength parameterization of the Davidenko equation or, more generally, equation (1.1) have been developed and tested in this work with the objective being to achieve both robustness and efficiency. A philosophy of this methodology, which aids considerably in the robustness, has been to follow the solution path closely wherein generally only two corrections back to the path are required. To achieve efficiency, the developed research code (ABCON) employs consecutive order Adams–Bashforth predictors to determine the stepsize by estimating the local truncation error in the lower-order formula, a comparison of three consecutive order methods to choose the order that yields the largest stepsize, and local extrapolation to improve accuracy. A key to the efficiency and robustness of these predictors, especially when combined with a Newton-like correction back to the solution path, is the approximation of arclength along the path at an accuracy commensurate with that of the tangents to the path.

The focus in this work has been on the predictor phase with an error stepsize control and not the development of a general predictor corrector-continuation algorithm. The method used in the computation of the tangent and the correction process is based on the bordering algorithm of Keller [17], which is quite general; however, the linear algebra used to solve the reduced problem is Gaussian elimination with scaled partial pivoting. Thus the test problems used for testing and comparison were small, dense but difficult ones taken from the papers of Watson, Billups, and Morgan [44], [45]; Heijer and Rheinboldt [9]; and Schwetlick and Cleve [36]. To compensate for this small class of problems and to give an indication of the potential performance in other problem classes, we have given performance measurements for several categories including the number of runs required to achieve success, the number of function evaluations, the number of linear systems solved, the number of matrix factorizations, and number of computed points on the path. ABCON has been tested and compared numerically with the state-of-the-art software packages HOMPACK (QF, NF, and DF) by Watson, Billups, and Morgan [45] and PITCON (Full and Chord) by Rheinboldt and Burkardt [31], [32]. Before summarizing these comparisons and conclusions, several caveats are in order. HOMPACK was designed for the solution of nonlinear equations by homotopy methods, particularly the regularizing homotopy, and the focus of PITCON was large sparse problems arising, for example, from practical engineering applications. Thus the performance measurements and test problems given in §7 are appropriate but not necessarily consistent with the design objectives of these latter two packages. Our objective has not been to show superiority of ABCON over HOMPACK and PITCON or to make any claim about their relative efficiency and robustness on their intended problem classes. The test set is much too small to extract

any such information. Instead, the objective has been to demonstrate the potential efficiency and robustness of the Adams–Bashforth predictors for predictor-corrector continuation methods. In the testing, HOMPACK and PITCON were not penalized for numbers they do not report; the extra work required by PITCON to compute the number of fold points was not counted (HOMPACK does not report these numbers); however, ABCON was penalized for not reporting all numbers correctly and the extra work required to determine the correct number of fold points was counted in the numbers for ABCON.

From the comparisons given in §7 and our further extensive experimentation with these variable order Adams–Bashforth predictors in the optimization setting [23], we can begin to make some general conclusions about the potential performances of these predictors. The successful use of a multistep method using more than two steps requires the stepsize in arclength along the path to be approximated with an accuracy commensurate with the accuracy required of the predictors. These higher-order Adams–Bashforth predictors, wherein a Newton-like correction back to the path is employed, require that one take relatively small and more accurate prediction steps, but this strategy can produce a more robust and efficient procedure, giving a more reliable trace of the path. As observed by Watson [45], an issue which is closely related to the latter point is that the general testing procedures used in evaluating the effectiveness of continuation procedures should account somehow for the total work done in solving a given problem, including the work involved in failed runs or runs used to determine optimal parameter settings. Indeed, ABCON performs exceedingly well in this regard, principally because the solution path is followed closely. Since the second correction is generally used only to verify the accuracy of the first full Newton correction in both ABCON Full and Chord, the full power of Newton's method is generally not useful in the correction process. Thus, in this methodology the chord method or a regular or sparse quasi-Newton method for the correction phase may be just as effective as a full Newton method on both small- and large-scale problems.

As discussed in the introduction and as reported in §7, the measurement of efficiency in these procedures is problem class dependent. If the number of matrix factorizations is the major concern, as it would be in the parametric study of nonlinear systems wherein one would encounter bifurcation and singularities, then ABCON Chord with its chord iteration appears to be more efficient than the remaining algorithms on these test problems. PITCON Chord was a close competitor and had a comparable success or failure rate. If the number of linear systems or the number of function evaluations is the gauge of efficiency, as might be appropriate for large scale systems wherein iterative methods are used for the linear systems, then one algorithm performed better than ABCON Chord, namely, PITCON Full, but it had the highest failure rate of all codes, nearly twice that of ABCON Chord and Full. Furthermore, PITCON Full used almost three times the number of matrix factorizations as ABCON Chord. PITCON Chord was, on the other hand, almost as efficient but its failure rate was slightly higher (2.0 versus 1.4) and it computed about half as many points. If we compare only those methods in which a full Newton method is used in the correction process, then the winners are ABCON Full for the number of points, HOMPACK NF for the number of Jacobian factorizations and the number of function evaluations, and PITCON Full for the number of linear systems solved; however, again ABCON Full (1.5) had nearly half the failure rate of HOMPACK NF (2.8) and PITCON Full (2.9). The only conclusion that we wish to extract from these numbers is that Adams–Bashforth predictors with the error-stepsize control developed in this work are potentially capable of adding efficiency to the robust procedure of following

the path closely, particularly for parameter continuation, and should be considered for general predictor-continuation procedures.

Another advantage of the Adams–Bashforth predictors is that, along with the predicted point on the curve as given by formula (2.5), there is the expression (2.8) for the predicted tangent, which has not been utilized in the current work since direct methods have been used for the solution of the linear systems. For problems using iterative methods, we could possibly utilize this predicted tangent to reduce the expense of the linear systems solved. Several other suggestions can be made for possible improvements in the efficiency of ABCON. One possibility is to factor the matrix only on the curve and to use the previous factored matrix in the correction process. This would give a more accurate tangent and, based on our experience, up to approximately 10 percent fewer steps along a given path. The recent theoretical work of Bourji and Walker [6] and Walker and Watson [41] and the numerical experience of Watson [43] suggest that the computation of both the corrected point and the tangent might be accomplished, at least for a limited number of steps, by using quasi-Newton updates instead of factoring the Jacobian at each step. This would certainly reduce the number of matrix factorizations, but may increase the number of linear systems solved.

Finally, Hermite predictors appear to be an attractive alternative to the Adams–Bashforth predictors used here since the orders increase by two as opposed to one per step. While two step Hermite predictors have been used effectively by various authors [36], [45], [49], our preliminary testing with variable order Hermite predictors has not been promising, possibly due to their greater sensitivity to errors in the tangents and computed arclength. However, if the use of quasi-Newton methods, as mentioned above, allows one to obtain a more accurate tangent at little extra expense, then the use of three-step (and higher) Hermite predictors may be efficient.

Acknowledgment. We wish to thank E. L. Allgower, K. Georg, L. T. Watson, and two referees for their constructive comments on an earlier version of this work. Time on a CDC Cyber 205 Supercomputer was provided by Colorado State University.

REFERENCES

[1] E. L. Allgower and K. Georg, *Predictor-corrector and simplicial methods for approximating fixed points and zero points of nonlinear mappings*, in Mathematical Programming: The State of the Art, A. Bachem, M. Grötschel, and B. Korte, eds., Springer-Verlag, New York, 1983.

[2] ———, *An Introduction to Numerical Continuation Methods*, Springer-Verlag, New York, 1990.

[3] J. H. Avila, *The feasibility of continuation methods for nonlinear equations*, SIAM J. Numer. Anal., 11 (1974), pp. 102–122.

[4] M. Berger and B. Gostiaux, *Differential Geometry: Manifolds, Curves and Surfaces*, Springer-Verlag, New York, 1988.

[5] W. E. Bosarge, *Iterative continuation and the solution of nonlinear two-point boundary value problems*, Numer. Math., 17 (1971), pp. 268–283.

[6] S. K. Bourji and H. F. Walker, *Least-change secant updates of nonsquare matrices*, Res. Report 38, Department of Mathematics, Utah State University, Logan, UT, 1987.

[7] T. Chan, *Techniques for large sparse systems arising from continuation methods*, in Numerical Methods for Bifurcation Problems, T. Kupper, H. Mittelmann, and H. Weber, eds., Birkhäuser, Boston, 1984, pp. 116–128.

[8] S.-N. Chow and J. C. Hale, *Methods of Bifurcation Theory*, Springer-Verlag, New York, 1982.

[9] C. Den Heijer and W. C. Rheinboldt, *On steplength algorithms for a class of continuation methods*, SIAM J. Numer. Anal., 18 (1981), pp. 925–948.

[10] J. E. DENNIS, JR. AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,* Prentice-Hall, Englewood Cliffs, NJ, 1983.

[11] P. DEUFLHARD, B. FIEDLER, AND P. KUNKEL, *Efficient numerical path following beyond critical points,* SIAM J. Numer. Anal., 24 (1987), pp. 912–927.

[12] P. DEUFLHARD, *A stepsize control for continuation methods and its special application to multiple shooting techniques,* Numer. Math., 33 (1979), pp. 115–146.

[13] J. J. DONGARRA, F. G. GUSTAVSON AND A. KARP, *Implementing linear algebra algorithms for dense matrices on a vector pipeline machine,* SIAM Rev., 26 (1984), pp. 91–112.

[14] K. GEORG, *Numerical integration of the Davidenko equation,* Lecture Notes in Mathematics 878, Springer-Verlag, New York, 1981, pp. 17–27.

[15] ———, *A note on stepsize control for numerical curve following,* in Homotopy Methods and Global Convergence, B. C. Eaves, F. J. Gould, H.-D. Peitgen, and M. J. Todd, eds., Plenum Press, New York, 1983, pp. 145–154.

[16] H. B. KELLER, *Numerical solution of bifurcation and nonlinear eigenvalue problems,* in Applications of Bifurcation Theory, P. H. Rabinowitz, ed., Academic Press, New York, 1977, pp. 359–384.

[17] ———, *Global homotopies and Newton methods,* in Recent Advances in Numerical Analysis, C. de Boor and G. H. Golub, eds., Academic Press, New York, 1978, pp. 73–94.

[18] ———, *The Bordering algorithm and path following near singular points of higher nullity,* SIAM J. Sci. Statist. Comput., 4 (1983), pp. 573–582.

[19] ———, *Numerical Methods in Bifurcation Problems,* Springer-Verlag, Berlin, 1987.

[20] F. T. KROGH, *Algorithms for changing the step size,* SIAM J. Numer. Anal., 10 (1973), pp. 949–965.

[21] T. KUPPER, H. D. MITTELMANN, AND H. WEBER, *Numerical Methods for Bifurcation Problems,* Birkhäuser-Verlag, Boston, 1984.

[22] T. KUPPER, R. SEYDEL, AND H. TROGER, EDS., *Bifurcation: Analysis, Algorithms, Applications,* Birkhäuser-Verlag, Boston, 1987.

[23] B. N. LUNDBERG, A. B. POORE AND B. YANG, *Smooth penalty functions and continuation methods for constrained optimization,* E. L. Allgower and K. Georg, eds., Lectures in Applied Mathematics, American Mathematical Society, Providence, RI, 1990.

[24] R. MEJIA, CONKUB: *A conversational path-follower for systems of nonlinear equations,* J. Comput. Phys., 63 (1986), pp. 67–84.

[25] R. MENZEL AND H. SCHWETLICK, *Parameterization via secant length and application to path following methods,* Numer. Math., 47 (1985), pp. 401–412.

[26] H. MITTELMANN, *A pseudo-arclength continuation method for nonlinear eigenvalue problems,* SIAM J. Numer. Anal., 23 (1986), pp. 1007–1016.

[27] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables,* Academic Press, New York, 1970.

[28] A. B. POORE AND K. AL-HASSAN, *The expanded Lagrangian system for constrained optimization problems,* SIAM J. Control Optim., 26 (1988), pp. 416–427.

[29] W. C. RHEINBOLDT, *Numerical analysis of continuation methods for nonlinear structural problems,* Comput. & Structures, 13 (1981), pp. 103–113.

[30] W. C. RHEINBOLDT AND J.V. BURKARDT, *A locally parameterized continuation process,* ACM Trans. Math. Software, 9 (1983), pp. 215–235.

[31] ———, *Algorithm* 596: *A program for a locally parameterized continuation process,* ACM Trans. Math. Software, 9 (1987), pp. 236–241.

[32] W. C. RHEINBOLDT, *On the solution of some nonlinear equations arising in the application of finite element methods,* in The Mathematics of Finite Elements and Applications II: MAFELAP 1975, J. R. Whiteman, ed., Academic Press, New York, 1976, pp. 465–482.

[33] ———, *Numerical Analysis of Parameterized Nonlinear Equations,* John Wiley, New York, 1986.

[34] R. D. RUSSELL AND J. CHRISTIANSEN, *Adaptive mesh selection strategies for solving boundary value problems,* SIAM J. Numer. Anal., 15 (1978), pp. 59–80.

[35] H. SCHWETLICK, *On the choice of steplength in path following methods,* Z. Angew. Math. Mech., 64 (1984), pp. 391–396.

[36] H. SCHWETLICK AND J. CLEVE, *Higher order predictors and adaptive steplength control in path following algorithms,* SIAM J. Numer. Anal., 24 (1987), pp. 1382–1393.

[37] L. F. SHAMPINE AND J. K. GORDON, *Computer Solution of Ordinary Differential Equations: The Initial Value Problem,* W. H. Freeman, San Francisco, CA, 1975.

[38] J. STOER AND R. BULIRSCH, *Introduction to Numerical Analysis,* Springer-Verlag, New York, 1980.
[39] A. H. STROUD AND D. SECREST, *Gaussian Quadrature Formula,* Prentice-Hall, NJ, 1966.
[40] H. WACKER, E. ZARZER, AND W. ZULEHNER, *Optimal stepsize control for globalized Newton method,* in Continuation Methods, H. Wacker, ed., Academic Press, New York, 1978.
[41] H. F. WALKER AND L. T. WATSON, *Least-change secant update methods for underdetermined systems,* Res. Report 41, Department of Mathematics, Utah State University, Logan, UT, 1988.
[42] L. T. WATSON, *An algorithm that is globally convergent with probability one for a class of nonlinear two-point boundary value problems,* SIAM J. Numer. Anal., 16 (1979), pp. 394–401.
[43] ———, *Numerical linear algebra aspects of globally convergent homotopy methods,* SIAM Rev., 28 (1986), pp. 529–545.
[44] L. T. WATSON, S. C. BILLUPS, AND A. P. MORGAN, *Algorithm* 652, HOMPACK: *A suite of codes for globally convergent homotopy algorithms,* ACM Trans. Math. Software, 13 (1987), pp. 281–310.
[45] L. T. WATSON, *A globally convergent algorithm for computing fixed points of C^2 maps,* Appl. Math. Comput., 5 (1979), pp. 297–311.
[46] H. A. WATTS, *The starting stepsize for an* O.D.E. *solver,* J. Comput. Appl. Math., 7 (1983), pp. 177–191.
[47] ———, *Step size control in ordinary differential equation solvers,* Trans. Soc. Comput. Simulation, 1 (1984), pp. 15–25.
[48] ———, HSTART—*An improved initial step size routine for* ODE *Codes,* Sandia Report SAND86-2633, Sandia National Laboratories, Albuquerque, NM, 1986.
[49] T. J. YPMA, *Following paths through turning points,* BIT, 22 (1982), pp. 368–383.