SIAM J. SCI. STAT. COMPUT.
Vol. 7, No. 3, July 1986

© 1986 Society for Industrial and Applied Mathematics
002

NUMERICAL SOLUTION OF NONLINEAR DIFFERENTIAL EQUATIONS WITH ALGEBRAIC CONSTRAINTS II: PRACTICAL IMPLICATIONS*

LINDA PETZOLD† AND PER LÖTSTEDT‡

Abstract. In this paper we investigate the behavior of numerical ODE methods for the solution of systems of differential equations coupled with algebraic constraints. Systems of this form arise frequently in the modelling of problems from physics and engineering; we study some particular examples from fluid dynamics and constrained mechanical systems. We investigate some of the practical difficulties of implementing variable-stepsize backward differentiation formulas for the solution of these equations, showing how to overcome problems of matrix ill-conditioning, and giving convergence tests and error tests which are supported by theory.

Key words. differential-algebraic, constraints, Lagrange multipliers, mechanical systems, high index systems

AMS(MOS) subject classification. 65L05

1. Introduction.

In this paper we investigate some of the practical difficulties involved with implementing backward differentiation formulas (BDF) for the solution of differential/algebraic equations (DAE) of the form

$$0=F_1(x,x',y,t),\qquad 0=F_2(x,y,t) \tag{1.1}$$

where the initial values of x and y are given at $t=0$ and $\partial F_1/\partial x'$ is nonsingular. Systems of this form arise frequently in the modelling of engineering problems, for example the simulation of electrical networks and mechanical systems, and the solution of the equations of fluid dynamics. In an earlier paper [1] we showed that for the systems under consideration (certain restrictive assumptions must be placed on (1.1); these assumptions are satisfied in many practical applications) the k-step constant stepsize BDF method converges to order of accuracy $O(h^k)$, where h is the stepsize. Here we are concerned with the practical difficulties such as varying the stepsize and dealing with ill-conditioned matrices which arise in implementing BDF methods for the solution of (1.1). Recently, similar systems of equations have been studied also by Brenan [2].

The idea of using BDF methods for systems of this type was introduced by Gear [3] and consists of replacing x' in (1.1) by a difference approximation, and then solving the resulting equations for approximations to x and y. Let $F=(F_1,F_2)^T$. To solve (1.1) numerically at t_n by the k-step BDF, we replace $x'(t_n)$ by $\rho x_n/h$ where ρ is the difference operator defined by

$$\rho x_n=\sum_{i=0}^{k}\alpha_i x_{n-i}, \tag{1.2}$$

$h=t_n-t_{n-1}$ and α_i are the BDF coefficients, to obtain the system of nonlinear equations

$$F\left(x_n,\frac{\rho x_n}{h},y_n,t_n\right)=0. \tag{1.3}$$

* Received by the editors July 19, 1984, and in revised form April 3, 1985.

† Computing Research and Development Division, Lawrence Livermore National Laboratory, Livermore, California 94550. The work of this author was supported by the U.S. Department of Energy Office of Basic Energy Sciences.

‡ Aerospace Division, SAAB-Scania, S-58188 Linköping, Sweden. The work of this author was supported in part by the National Swedish Board for Technical Development when the author was at the Department of Numerical Analysis and Computing Science, Royal Institute of Technology, S-10044 Stockholm, Sweden.

This system has a unique solution [4] if the inverse of the scaled Jacobian matrix

$$hJ_n = \begin{pmatrix} \alpha_0 \frac{\partial F_1}{\partial x'} + h\frac{\partial F_1}{\partial x} & h\frac{\partial F_1}{\partial y} \\ h\frac{\partial F_2}{\partial x} & h\frac{\partial F_2}{\partial y} \end{pmatrix} \tag{1.4}$$

exists.

The types of equations that we consider here arise commonly in several areas of application, which are discussed in more detail in [5]. For example, the flow of an incompressible, viscous fluid is described by the Navier-Stokes equations

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\nabla p + \gamma \nabla^2 u, \tag{1.5a}$$

$$\nabla \cdot u = 0 \tag{1.5b}$$

where u is the velocity in two or three dimensions, p is the pressure, and γ is the kinematic viscosity. After spatial discretization of (1.5) with a finite difference or finite element method, the vectors U and P, approximating u and p, satisfy [6]

$$M\dot{U} + (K + N(U))U + CP = f(U, P), \tag{1.6a}$$

$$C^T U = 0 \tag{1.6b}$$

which has the form (1.1). The mass matrix M is the identity matrix (finite differences) or a symmetric positive definite matrix (finite elements). The discretization of the operator ∇ is C and the forcing function f emanates from the boundary conditions.

Another application which fits into this general framework is the simulation of mechanical systems of rigid bodies interconnected directly by joints or via other components such as springs and dampers. The vector q of coordinates of the bodies satisfies the following equations [7]

$$M(q)q'' = f(q, q', t) + G(q)\lambda, \tag{1.7a}$$

$$\Phi(q) = 0. \tag{1.7b}$$

The mass matrix M is nonsingular almost everywhere, λ is the Lagrange multiplier vector and $\partial\Phi/\partial q = G^T$. The algebraic equation (1.7b) often represents geometrical constraints on the system. A simple example of a system such as (1.7) is the physical pendulum. Let L denote the length of the bar, λ is proportional to the force in the bar, and x and y the Cartesian coordinates of the infinitesimal ball of mass one in one end of the bar. Then x, y and λ solve the DAE system

$$\begin{aligned} x'' &= \lambda x, \\ y'' &= \lambda y - g, \\ 0 &= \tfrac{1}{2}(x^2 + y^2 - L^2), \end{aligned} \tag{1.8}$$

where g is the gravity constant.

To state our results, we must first introduce the concept of the *index* of a DAE system. The index is a measure of the singularity of a system. Standard form ODEs, $y' = f(t, y)$, have index zero, the fluid flow system (1.6) has index two, and the constrained mechanical system (1.7) has index three. In general, the higher the index, the more severe the numerical difficulties that we can expect. For the purposes of this paper we will simply define the index as the number of times the constraints of the system must be differentiated in order to obtain a standard form ODE system. This definition is compatible with previous definitions, for the systems considered here [8].

To illustrate the idea of index, we compute the index of the mechanical system (1.7). If the initial condition $q_0 = q(0)$ is consistent with (1.7b), $\Phi(q_0) = 0$, then the algebraic constraint (1.7b) can be replaced by its differentiated form

$$G^T(q)q' = 0. \tag{1.9}$$

If $\Phi(q_0) = 0$, $q_0' = q'(0)$ and $G^T(q_0)q_0' = 0$ then the condition (1.9) is equivalent to its derivative,

$$\frac{d}{dt}(G^T q') = G^T q'' + G'^T q' = 0. \tag{1.10}$$

We obtain a system of linear equations satisfied by λ by introducing q'' from (1.7a) into the expression above and solving for λ,

$$\lambda = -(G^T M^{-1} G)^{-1}(G^T M^{-1} f + G'^T q'). \tag{1.11}$$

This condition replaces (1.9). Finally, if λ_0, q_0, q_0' satisfy (1.11) at $t = 0$ we can differentiate (1.11) once more and the resulting equation, coupled with (1.7a), form a standard-form ODE system. The index of the original system (1.7) is three, because the constraint was differentiated three times to obtain a standard form ODE system. Similarly, it is easy to verify that the index of the equivalent system (1.7a, 1.9) is two, and that the index of the fluid dynamics system (1.6) is two [5].

In [1], [5], we showed that the k-step constant-stepsize BDF method converges to order $O(h^k)$ for systems of the form (1.1) where the initial values are consistent and the functions are sufficiently smooth, provided the system satisfies

ASSUMPTION 1.1.

(1) The index is less than or equal to one, or

(2) the index is equal to two and $\partial F_2/\partial y \equiv 0$, or

(3) the index is equal to three and the system has the form (1.7).

These conditions are satisfied by the fluid dynamics and mechanical systems mentioned earlier. In this paper we study the practical difficulties which are inherent in implementing a variable-stepsize code based on formulas such as BDF for solving these types of problems.

Often, as in the case of the simulation of mechanical systems of rigid bodies, a DAE system may be written in a different but analytically equivalent way. In some cases this is a good idea, because it may reduce the index and make the problem easier to solve by numerical methods. Therefore, in § 2 we discuss some analytical techniques for rewriting DAE systems in a simpler form, and the reasons why we may or may not want to do this.

The remainder of the paper is concerned with the numerical difficulties associated with solving the systems in their original, high index form. A difficulty that is common to the solution of all high index DAE systems is that the iteration matrix which is used by numerical ODE methods is poorly conditioned when the stepsize is small. This can cause variable-stepsize codes to fail or to give poor diagnostics in case of failure from other causes [9]. In § 3 we give a general technique for scaling the equations and variables in (1.1) that circumvents this difficulty. Scaling can also cause trouble with the convergence and error tests in an automatic code. We study these problems in §§ 4 and 5 and devise convergence tests and error tests that do not suffer from these difficulties, and that are justified by the theory in [1].

2. Alternative forms for DAE systems. In this section we consider some techniques for rewriting a system in an alternative form that may be easier to solve numerically. All of the different forms of the equations that we consider are analytically equivalent in the sense that, given a consistent set of initial conditions, different forms of a system have the same analytical solution. Computationally, however, some forms of the equations may have much different properties than others. We discuss some of the advantages and disadvantages of rewriting a high index DAE system in a different form.

As an example, let us consider some different ways to solve the constrained mechanical system (1.7). First of all, we can attempt to solve the system in its original, index-three form, using an implicit numerical method such as BDF. This technique is actually used in some codes [10], [11] for solving mechanical systems. Solving the problem in this way has the advantages that it is easy to formulate the system (we do not have to differentiate the constraints or rewrite the system in any way), the sparsity of the system is preserved, and the constraints are satisfied exactly on every step. However, there are several difficulties in using a variable-stepsize BDF code for solving systems in this form. First, the iteration matrix that the code uses at each time step is very ill-conditioned for small stepsizes. This can cause severe difficulties with the Newton iteration and with stepsize selection in the code. This difficulty can be partially remedied via the scaling techniques considered in § 3. Secondly, error estimation and stepsize selection algorithms that are normally used in variable-stepsize codes fail for this type of problem (see §§ 4 and 5). One way to overcome this difficulty is to base the error control and stepsize selection strategies only on the vector q, and not on the velocities q' or the Lagrange multipliers λ. We can use the theory developed in [1] to show that the constant-stepsize BDF converges for problems written in this form, but this says nothing about what will happen when there are errors, for example, in the initial velocities. We have found from experiment that a variable-stepsize code which bases its stepsize selection and order control strategies only on q can obtain completely wrong answers when the initial velocities fail to satisfy the condition that the derivative of the constraint should be zero. Thus, we must be very careful that the initial conditions are consistent in the sense that not only the constraint, but also the derivative of the constraint, is zero at the initial time. Unfortunately, all of the techniques for solving mechanical systems that we discuss in this section experience some type of serious difficulty when the initial conditions to the original problem are not chosen to be consistent. For this reason, we do not reject this method (of solving the original index-three problem with variable-stepsize BDF) entirely, but we do doubt its reliability when the initial conditions are inconsistent, or in situations where there are steep gradients or discontinuities in the velocities. It may be possible to reliably solve systems in the index-three form with methods other than BDF, such as extrapolation [8] or defect correction, by controlling errors on the velocities as well as the positions, but we will not take up this subject further in this paper.

A second way of solving (1.7) is to differentiate the constraint and solve the system (1.7a), (1.9). This approach produces a poorly conditioned iteration matrix (though not as poorly conditioned as solving the system in its original form), but we can again eliminate most of these difficulties by scaling the problem as described in § 3. Stepsize selection and error control strategies for variable-stepsize BDF codes fail for these problems, unless we somehow exclude the Lagrange multipliers λ from the error control decisions. In contrast to the error control strategies proposed in the previous paragraph, there is some justification (see § 5) for excluding λ from the error control decisions. In this approach, it is a simple task to verify that the initial conditions $q'(0)$ on the velocity satisfy the algebraic constraint (1.9). If $G^T(q(0))q'(0) \neq 0$ then $q'(0)$

can be corrected by computing an impulse Λ in the system at $t=0$ such that

$$q'_+ - q'(0) = G\Lambda, \qquad G^T q'_+ = 0.$$

However, now we must be careful that the constraint itself is satisfied at the initial time. An initial error in the constraint will contaminate the solution in the whole time interval of interest. There are, though, reasons to believe [5] that any system which does not satisfy the constraint initially is not a good physical model. The main difficulty with solving (1.7a), (1.9) is that of "drifting off" the original constraint. (Note that this is not a problem for solving the Navier-Stokes equations (1.6), which are essentially the same form as (1.7a), (1.9) because there the constraint that we are using is the original constraint of the system.) This formulation of the problem does not force the constraint to be satisfied on every step, and there may be a tendency for the amount by which the constraint is not satisfied to increase from step to step. By using small stepsizes (in an automatic code, by keeping the error tolerances fairly stringent), we can keep small these errors in the amount by which the constraint is not satisfied. Whether this is a serious problem or not depends on the application, although clearly it could be troublesome if the solution is desired over a long interval in time.

A third strategy, which is used in some codes for solving mechanical systems [10], is to eliminate the Lagrange multipliers analytically by means of methods in analytical mechanics to obtain a standard form ODE system. The system of ODEs is assembled from a data structure describing the mechanical system. If the resulting problem is not stiff, this approach has the advantage that the system can be solved by an explicit numerical method. The number of unknowns after this type of transformation usually is smaller, but the sparsity of the system has decreased, which is an important consideration if the problem happens to be stiff. Again, we must be very careful that the initial conditions satisfy that both the constraint and the derivative of the constraint are zero, or we will obviously obtain a solution which is nonsense. A constraint corresponding to an eliminated Lagrange multiplier is automatically satisfied in the chosen representation of the mechanical system. Consider the pendulum (1.8) as an example. Let

$$x = L\sin\varphi, \qquad y = -L\cos\varphi.$$

Then the algebraic constraint of constant length is fulfilled and the well-known ODE is

$$\ddot{\varphi} - \gamma\sin\varphi = 0, \qquad \gamma = \frac{g}{L}.$$

Various combinations of the above strategies can be employed. Baumgarte [12] discusses a technique for circumventing this problem of "drifting off" the constraints $\Phi(q)$ by adding to the original equations an equation consisting of a linear combination of Φ, $d\Phi/dt$ and $d^2\Phi/dt^2$. The linear combination is chosen so that the resulting system damps errors in satisfying the constraint equation. This approach is similar, but not identical, to penalty function methods (Lötstedt [13], Sani et al. [14]). Depending on the choice of the parameters in the linear combination, we may see any of the difficulties discussed earlier. This technique introduces extraneous eigenvalues into the system, which may or may not cause difficulties. Finally, the penalty techniques have the disadvantage that if the initial conditions are not posed correctly, they introduce a nonphysical transient into the problem [14].

While all of these techniques have their shortcomings, we have attempted here to put them on a firmer foundation, and to suggest some ways to implement them more effectively. All of these techniques experience serious difficulties when the initial conditions are not consistent.

Before leaving the subject of alternate forms for DAE systems, there is one more aspect of this problem that we wish to consider. Sometimes there is a choice of which variables to use for solving a problem. For example, in the system

$$\begin{aligned} u' &= v, \\ v' &= f(u, v, t) + G(u)\lambda, \\ G^T v &= 0 \end{aligned} \tag{2.1}$$

we could have replaced v in the constraint by u' to obtain

$$\begin{aligned} u' &= v, \\ v' &= f(u, v, t) + G(u)\lambda, \\ G^T u' &= 0. \end{aligned} \tag{2.2}$$

Are there any advantages in writing the system in one form over the other? Using BDF with Newton iteration for linear problems in exact arithmetic, the two forms of the equations give identical solutions. (This is because Newton's method is exact in one iteration for linear systems, and because the equations which result from discretizing both systems by BDF are identical—this last fact is obviously true for nonlinear systems too.) For nonlinear systems in exact arithmetic we know of no reasons why Newton's method would be more likely to converge for one form of the equations than the other. Both forms of the system may lead to very poorly conditioned iteration matrices. In the next section we will suggest scaling techniques to overcome this difficulty. These techniques are directly applicable to problems of the form (2.1) but not to (2.2) (though it is often possible to devise ways to scale the analytically equivalent systems such as (2.2)). This may be a reason to prefer (2.1). Everything that we discuss in this paper, with the exception of the scaling techniques introduced in § 3, is applicable to systems such as (2.2) where these obvious substitutions have been made. Thus, if it is more convenient to solve one of these alternative forms of a system, then there is some justification for doing so.

3. Conditioning of matrices arising in the solution of DAEs. In this section we study the solution of DAEs of the form (1.1). Conditioning is a problem for DAE systems, and especially for high index systems, because the condition number of the iteration matrix for a system with index of m is $O(h^{-m})$ ([5, Thm. 4.1]). We describe schemes for scaling systems (1.1) so that the iteration matrices are no longer singular as $h \to 0$, and we discuss how these scaling techniques can be conveniently implemented into existing DAE software.

The system of linear equations to be solved in each Newton iteration step is

$$Az = b.$$

If we use Gaussian elimination with partial pivoting we know that the computed solution $z + \Delta z$ satisfies

$$(A + \Delta A)(z + \Delta z) = b, \tag{3.1}$$

where

$$\|\Delta A\|_\infty \leqq r\mathbf{u}\|A\|_\infty. \tag{3.2}$$

In (3.2) r is a moderate number and $\mathbf{u}$ is the machine unit. The accuracy of this computation is improved if we introduce a row scaling of A by premultiplying by a diagonal matrix D. Then by (3.2)

$$\begin{aligned}|\Delta z_i| &\leqq |(A^{-1}D^{-1}D\Delta A(z+\Delta z))_i|\\ &\leqq \sum_j |(A^{-1}D^{-1})_{ij}|\,|(D\Delta A(z+\Delta z))_j|\\ &\leqq \sum_j |(A^{-1}D^{-1})_{ij}|\,\|D\Delta A\|_\infty \|z+\Delta z\|_\infty\\ &\leqq r\mathbf{u}\sum_j |(A^{-1}D^{-1})_{ij}|\,\|DA\|_\infty\|z+\Delta z\|_\infty.\end{aligned} \tag{3.3}$$

We consider three cases: index one, index two, and index three systems of the form

$$\begin{aligned} x'-f(x,y,t)&=0,\\ g(x,y,t)&=0.\end{aligned} \tag{3.4}$$

(These ideas extend easily to the slightly more general form (1.1).) For these problems, the iteration matrix is written as

$$hJ_n=\begin{bmatrix}\alpha_0 I-h\dfrac{\partial f}{\partial x} & -h\dfrac{\partial f}{\partial y}\\ h\dfrac{\partial g}{\partial x} & h\dfrac{\partial g}{\partial y}\end{bmatrix}. \tag{3.5}$$

Case I. When the index is one, we have that $\partial g/\partial y$ is nonsingular, so hJ_n is nonsingular as $h\to 0$ if we scale the rows corresponding to the algebraic constraint by $1/h$. Since we are not scaling variablés, but only equations, the effect of this scaling should be to improve the accuracy of the solution of the linear system, for *all* variables.

Case II. For this case, we will assume that the index is two, and that $\partial g/\partial y\equiv 0$. By explicitly computing $(hJ_n)^{-1}$ we find that the orders of the blocks of the inverse are

$$\begin{pmatrix}1 & 1/h\\ 1/h & 1/h^2\end{pmatrix}, \tag{3.6}$$

where the elements in the first row correspond to x and those in the second row to y. If we scale the bottom rows of hJ_n (corresponding to the "algebraic" constraints) by $1/h$, then the scaled matrix can be written as

$$hJ'_n=\begin{pmatrix}\alpha_0 I-h\dfrac{\partial f}{\partial x} & -h\dfrac{\partial f}{\partial y}\\ \dfrac{\partial g}{\partial x} & 0\end{pmatrix}. \tag{3.7}$$

It follows from (3.3) that roundoff errors proportional to $\mathbf{u}/h$ and $\mathbf{u}/h^2$ are introduced in x and y, respectively, while solving the unscaled linear system. With the suggested scaling the roundoff errors are of $O(\mathbf{u})$ in x and $O(\mathbf{u}/h)$ in y. As $h\to 0$ these errors can begin to dominate the solution y. This is likely to cause difficulties for the error estimates and convergence tests in an automatic code. These difficulties, along with what can be done to minimize their effects, will be described in greater detail later. For now, we merely note that the effect of the proposed scaling is to control the size of the roundoff errors in x which are introduced in solving the linear system. At the same time, the "algebraic" variables y may contain errors proportional to $\mathbf{u}/h$. However, since the values of y do not affect the state of the system directly (that is, how the system will respond at future times), we may be willing to tolerate much larger errors in y than in x. In any case, this scaling is a significant improvement over the original scaling (3.5). For the scaled system, the errors are considerably diminished, and the largest errors are confined to the variables which are in some sense the least important. Painter [15] describes difficulties due to ill-conditioning for solving incompressible Navier–Stokes equations of the form (1.6), and employs essentially the same scaling that we have suggested here to solve the problem. These difficulties are most severe when an automatic code is using a very small stepsize, as in starting a problem or passing over a discontinuity in some derivative.

We further note that if we are using Gaussian elimination with partial pivoting, we do not need any column scaling. The solution will be the same without this scaling, because it does not affect the choice of pivots [16]. What the analysis shows is that the errors which are due to ill-conditioning are concentrated in the "algebraic" variables of the system and not in the "differential" variables. Thus, we must be particularly careful about using the "algebraic" variables in other tests in the code which might be sensitive to the errors in these variables.

Case III. For this case, we will work with systems of the form

$$\begin{aligned} &u' = v, \\ &v' = f(u, v, t) + G(u)\lambda, \\ &\Phi(u) = 0, \end{aligned} \tag{3.8}$$

i.e., mechanical systems where we have assumed, without loss of generality for the purposes of this discussion, that $M = I$. For these systems, the iteration matrix is written as

$$hJ_n = \begin{pmatrix} \alpha_0 I & -hI & 0 \\ hX & \alpha_0 I + hY & -hG \\ hG^T & 0 & 0 \end{pmatrix} \tag{3.9}$$

where

$$X = -\frac{\partial f}{\partial u} - \frac{\partial G}{\partial u}\lambda,$$

and

$$Y = -\frac{\partial f}{\partial v}.$$

Let $(hJ_n)^{-1}$ be partitioned into nine blocks such that the three block rows correspond to the variables u, v and λ and the three block columns correspond to the differential equations and the algebraic constraint in (3.8). Then the leading terms in $\gamma = h/\alpha_0$ in the blocks of $(hJ_n)^{-1}$ are

$$\frac{1}{\alpha_0}\begin{pmatrix} I-P & \gamma(I-P)S^{-1} & \gamma^{-1}S^{-1}GA \\ -\gamma^{-1}P & (I-P)S^{-1} & \gamma^{-2}S^{-1}GA \\ -\gamma^{-2}AG^T & -\gamma^{-1}AG^TS^{-1} & \gamma^{-3}A \end{pmatrix}. \tag{3.10}$$

In (3.10) the notation is $S = I + \gamma Y + \gamma^2 X$, $A = (G^T S^{-1} G)^{-1}$ and $P = S^{-1}GAG^T$. If we perform row equilibrium, i.e. scale the last row in (3.9) by $1/h$, then (3.3) and (3.10) give that the roundoff errors in u, v and λ are proportional to $\mathbf{u}$, $\mathbf{u}/h$ and $\mathbf{u}/h^2$, respectively. Note that if we scale the second row in (3.9) by h then the ith row block in each column of $(hJ_n)^{-1}D^{-1}$ is of $O(\gamma^{1-i})$. The errors with either of these scalings are much smaller than if we had solved the system with the original unscaled matrix (3.9), which had condition number $O(1/h^3)$. We may not be interested in the values of λ, because these Lagrange multipliers have no direct effect on the state of the system. The situation is not quite so simple with respect to the velocities v, however. The components of v in certain directions *do* in part determine the state of the system. In the two-dimensional pendulum example (1.8) such a direction is perpendicular to the bar. However, the $O(\mathbf{u}/h)$ errors that we can expect in v using this scaling are still considerably smaller than the $O(\mathbf{u}/h)$, $O(\mathbf{u}/h^2)$ and $O(\mathbf{u}/h^3)$ errors that we could expect in u, v and λ in solving the original unscaled system. The analysis shows that the errors which are due to ill-conditioning are concentrated in the variables v and λ, and not in u. Thus, we must be careful about using the "algebraic" variables v and λ in other tests in the code.

One further question that remains is how to implement row scaling in a general DAE code. There is a very nice solution to this problem. In a general purpose DAE code [18] (for solving systems of the form $F(t, y, y') = 0$), there is a subroutine which the user writes for computing the residual $F(t, y, y') = \Delta$, given (t, y, y'). The user can scale Δ inside this subroutine, and according to our guidelines, if we pass the stepsize h in the argument list to this subroutine. This way, the scaling costs virtually nothing. An alternative idea is to provide an option to automatically do row equilibration, or to use linear system solvers which perform row scaling, as suggested in Shampine [17].

4. Tests for terminating the corrector iteration.

In the last section we saw that for high index systems, even with scaling, there are relatively large errors in some of the variables. For the most part, these variables do not determine the state of the system, so these errors are in some sense tolerable. However, from the point of view of an automatic code where we must have some criterion for deciding when to terminate the corrector iteration, the errors in these variables are still a source of difficulties. Our objective in this section is to show that, from the point of view of propagation of errors in the state variables of a system, it is sufficient to terminate the Newton iteration based on the errors in the *scaled* variables, where the variables are scaled by powers of the stepsize as in the previous section. This eliminates troubles due to ill conditioning in the corrector iteration part of a code.

We will examine the propagation of errors caused by the interruption of the Newton iterations for a BDF method, for systems of the form (1.1) of index one, two and three.

For these purposes, let (x_n, y_n) be the computed solution (where the corrector iteration has not necessarily been solved exactly), and let $(\tilde{x}_n, \tilde{y}_n)$ be the true solution to the difference equation. Then $\tilde{x}_n = x_n + \delta_n^x$, $\tilde{y}_n = y_n + \delta_n^y$, where δ_n^x and δ_n^y are the errors in x_n and y_n due to terminating the Newton iteration early. Let $e_n^x = x_n - x(t_n)$, $e_n^y = y_n - y(t_n)$ be the global errors and let τ_n be the local truncation error. Then we have

$$\begin{aligned}
0 &= F(\tilde{x}_n, \rho\tilde{x}_n/h, \tilde{y}_n, t_n)\\
&= F(x_n + \delta_n^x, \rho(x_n + \delta_n^x)/h, y_n + \delta_n^y, t_n)\\
&= F(x(t_n) + e_n^x + \delta_n^x, \rho(x(t_n) + e_n^x + \delta_n^x)/h, y(t_n) + e_n^y + \delta_n^y, t_n)\\
&= F(x(t_n), x'(t_n), y(t_n), t_n) + \frac{\partial F}{\partial x}(e_n^x + \delta_n^x)\\
&\quad + \frac{\partial F}{\partial x'}(\rho(e_n^x + \delta_n^x)/h + \tau_n)\\
&\quad + \frac{\partial F}{\partial y}(e_n^y + \delta_n^y) + (\text{higher order terms}).
\end{aligned} \tag{4.1}$$

For notational convenience, introduce $F_{1x}=\partial F_1/\partial x$, $F'_{1x}=\partial F_1/\partial x'$, $A_1=\alpha_0 F'_{1x}+hF_{1x}$, $A_2=\partial F_1/\partial y$, $A_3=\partial F_2/\partial x$, $A_4=\partial F_2/\partial y$. Then the errors must satisfy

$$(4.2)\qquad \begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix}\begin{pmatrix} e_n^x+\delta_n^x \\ e_n^y+\delta_n^y \end{pmatrix}=\begin{pmatrix} -F'_{1x}(c_n+h\tau_n) \\ 0 \end{pmatrix}+(\text{higher order terms})$$

where $c_n=\sum_{i=1}^k \alpha_i(e_{n-i}^x+\delta_{n-i}^x)$. Only higher order terms of δ_n^x and δ_n^y are present in the right-hand side of (4.2). From (4.2) we find that

$$(4.3)\qquad \begin{pmatrix} e_n^x \\ e_n^y \end{pmatrix}=\begin{pmatrix} -\delta_n^x \\ -\delta_n^y \end{pmatrix}+\begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix}^{-1}\left[\begin{pmatrix} \zeta \\ 0 \end{pmatrix}+(\text{higher order terms})\right],$$

where ζ is the right-hand side of (4.2a). For the purposes of only controlling the errors in x and y at t_n, the same criterion can be used on both δ_n^x and δ_n^y. In the analysis in [1] of the accumulated error in x and y as time progresses we derived the requirements on the residual η from the Newton iteration to obtain $\|e_n^x\|=O(h^k)$ and $\|e_n^y\|=O(h^k)$ with the kth order BDF. It follows from (4.1) that

$$0=hF(x_n,(\alpha_0 x_n+c_n)/h,y_n,t_n)+\begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix}\begin{pmatrix} \delta_n^x \\ \delta_n^y \end{pmatrix}+(\text{higher order terms}).$$

Hence,

$$(4.4)\qquad \begin{pmatrix} A_1 & hA_2 \\ hA_3 & hA_4 \end{pmatrix}\begin{pmatrix} \delta_n^x \\ \delta_n^y \end{pmatrix}\approx h\eta=-h\begin{pmatrix} F_{1n} \\ F_{2n} \end{pmatrix},$$

where F_{1n} and F_{2n} are evaluated at x_n, y_n and t_n. In the index one case η must be of $O(h^k)$ according to the results in [1]. The iteration should be terminated when δ_n^x and $h\delta_n^y$ are of $O(h^{k+1})$ in (4.4). Let $\eta^T=(\eta_1^T,\eta_2^T)$. If $A_4\equiv 0$ and the index is two, then η_1 and η_2 must be of $O(h^k)$ and $O(h^{k+1})$. This is achieved by letting δ_n^x and $h\delta_n^y$ be proportional to h^{k+1} in (4.4). Equation (4.4) is satisfied with $x^T=(u^T,v^T)$ and $y=\lambda$ by the index three system (3.8). Here, we require $\|\eta_1\|=O(h^{k+1})$ and $\|\eta_2\|=O(h^{k+2})$ in [1] which is obtained by taking $\|\delta_n^x\|=O(h^{k+1})$ and $\|h\delta_n^y\|=O(h^{k+1})$. These conclusions are independent of the row scaling.

In general, variables should never be totally excluded from the test for convergence of the corrector iteration. Even if a variable occurs linearly, as for example in Brown and Gear [19], it is a mistake to exclude it from the convergence test in an automatic code. The reason for this is that codes are so often using Jacobians which were calculated on previous steps and possibly even based on different stepsizes that in general we cannot count on the Newton iteration to converge in one step.

5. Error tests.

In this section we will examine the reasons why some variables should be excluded from the error test in an automatic code.

It follows from (4.1) and (4.2) that the errors in the "algebraic" variable y on previous time steps, e_i^y, $i<n$, do not directly influence the errors in any of the variables at the current time t_n for systems of index one and two satisfying Assumption 1.1 since they do not appear in (4.1). Therefore, we can consider deleting these variables from the error estimate. This could be advantageous for the smooth operation of a code. Consider first the case of solving index one systems. Suppose, for example, that on one step we make a fairly large mistake by terminating the Newton iteration before it has really converged, and that on this step the value of y that should have been computed has a large error in it, but that based on the incorrect value it passes the error test anyway. If we base the error test on y, then on the next step this could cause a big problem, because the new value of y does not approach the old (incorrect) value of y, so that their difference, and hence the error estimate, does not approach zero as

$h \to 0$. Because of this, we feel that in this case it is probably wise to leave y out of the error control decisions. The main drawback in this strategy is that if we would like to know the values of y at interpolated points (between mesh points) then the stepsize should be based in part on the values of y. For index two systems, the stepsize control strategy in an automatic BDF code will fail, for reasons explained in Petzold [9], unless the "algebraic" variables y are excluded from the error test. For Navier-Stokes systems, this means that the pressure should be excluded from the error test, and for constrained mechanical systems (1.7a), (1.9) where the constraint has been differentiated once, it means that the Lagrange multipliers should be excluded from the error test.

Another possible error estimate is based on the observation that the part of the error in x and y due to the truncation error $h\tau_n$ in (4.2) in the present step is

$$e_{\tau_n} = \begin{pmatrix} e^x_{\tau_n} \\ e^y_{\tau_n} \end{pmatrix} = -(hJ_n)^{-1} \begin{pmatrix} F'_{1x} h\tau_n \\ 0 \end{pmatrix}, \tag{5.1}$$

where higher order terms have been omitted. This is the contribution to the global error that is directly influenced by a change of the stepsize. An estimate of e_{τ_n} can be obtained by multiplying the usual error estimate by the matrix

$$(hJ_n)^{-1} \begin{pmatrix} F'_{1x} & 0 \\ 0 & 0 \end{pmatrix}.$$

An approximate factorization of hJ_n is available from the Newton iteration. Then an estimate of e_{τ_n} can be computed according to (5.1). The step size is chosen such that e_{τ_n} is less than a given error tolerance. This is a generalization of the estimate given by Sincovec et al. [20] for constant-coefficient DAE systems (see also Petzold [9]). If $A_4 = 0$ and the index is two, we find from (3.6) that $e^x_{\tau_n} \sim h\tau_n \sim h^{k+1}$ as we would expect but that $e^y_{\tau_n} \sim \tau_n \sim h^k$.

For the index three mechanical systems, the situation is almost as simple. According to (3.10) the leading term in the contribution from the discretization error $h\tau_n$ to the errors e^u_n and e^v_n is

$$\begin{pmatrix} e^u_{\tau_n} \\ e^v_{\tau_n} \end{pmatrix} = \frac{1}{\alpha_0} \begin{pmatrix} I-P & \gamma(I-P)S^{-1} \\ -P/\gamma & (I-P)S^{-1} \end{pmatrix} \begin{pmatrix} h\tau^u_n \\ h\tau^v_n \end{pmatrix}. \tag{5.2}$$

The next term of higher order in the lower left-hand block is of $O(\gamma)$. The lower left part of the first matrix above is of $O(h^{-1})$. In general we have $P\tau^u_n \neq 0$ and the order of the asymptotic behavior of $e^v_{\tau_n}$ is one less than what we can expect for differentiated variables from index one or two systems. What we would ideally like to control is $e^u_{\tau_n}$ and the truncation error in the components of the velocity vector v which are in allowable directions (which would not cause u to violate the constraint). Stepsize control strategies used in BDF codes will fail [9] if we base the strategy on the values of v and/or λ. One way to solve this problem using a general purpose BDF code is to base the error control and stepsize selection strategies solely on u. This is actually done in a production code [10], [11] for solving mechanical systems. However, we question whether this is always a reliable procedure.

Another possible error estimate for the index three mechanical system is based on the observation from (5.2) that we can obtain the leading terms in $e^u_{\tau_n}$ and $e^v_{\tau_n}$ by solving

$$(hJ_n) \begin{pmatrix} e^u_{\tau_n} \\ e^v_{\tau_n} \\ z \end{pmatrix} = \begin{pmatrix} h\tau^u_n \\ h\tau^v_n \\ 0 \end{pmatrix}$$

where the value of z does not matter to us. Now if we are only interested in the components of v in the nullspace $N(G^T)$ of G^T then such a component is

$$v^* = (I-P)v, \tag{5.3}$$

where

$$P = S^{-1}G(G^TS^{-1}G)^{-1}G^T,$$

as in (3.10). The matrix P is a projector [21] since $P^2 = P$. Let $\|\cdot\|$ denote the spectral matrix norm. For h sufficiently small

$$S^{-1} = I + hS_1, \qquad \|S_1\| = O(1),$$

and from the definition of P in (5.3)

$$P = P_0 + hP_1, \qquad \|P_1\| = O(1),$$

where $P_0 = G(G^TG)^{-1}G^T = P_0^T$, an orthogonal projector [21]. Therefore,

$$\|P\| = 1 + O(h), \qquad \|I-P\| = 1 + O(h),$$

and $\|v^*\|$ is bounded by $\|v\|$,

$$\|v^*\| \leqq \|v\|(1+O(h)). \tag{5.4}$$

From the pendulum example in (1.8) and the definition of S and P we find that with $u^T = (x, y)$

$$\|S-I\| = O(h^2), \quad P_0 = u(u^Tu)^{-1}u^T, \quad \|P-P_0\| = O(h^2).$$

The motion of the pendulum ball is constrained in the direction $u = P_0u$ and free in the perpendicular direction $w^T = (y, -x)$ for which $P_0w = 0$.

The contribution to the error in v^* from the truncation error is

$$e_{\tau_n}^{v^*} = (I-P)e_{\tau_n}^{v}. \tag{5.5}$$

It follows from (3.10) that the leading term in $e_{\tau_n}^{v^*}$ is the solution to

$$(hJ_n)\begin{pmatrix} e_{\tau_n}^{v^*} \\ z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} e_{\tau_n}^{v} \\ 0 \\ 0 \end{pmatrix}.$$

The reasons why the particular projector in (5.3) is chosen are first that P is close to an orthogonal projector so that a relation such as (5.4) is satisfied and second the determination of $e_{\tau_n}^{v^*}$ is not too complicated. Thus we can obtain $e_{\tau_n}^{u}$ and $e_{\tau_n}^{v^*}$ with two extra back substitutions using the approximate factorization of hJ_n which was computed during the Newton iteration. It follows from (5.2) and (5.5) that the terms of lowest order in $e_{\tau_n}^{v^*}$ are

$$(I-P)(-hP\tau_n^u/\gamma + h(I-P)S^{-1}\tau_n^v) = h(I-P)\tau_n^v + O(h^{k+2}).$$

The projector in front of the truncation errors is bounded independently of h. Hence, $e_{\tau_n}^{v^*}$ is proportional to h^{k+1}. This is in contrast to the error $e_{\tau_n}^{v}$ which is $O(h^k)$ due to the error component $Pe_{\tau_n}^{v} \sim P\tau_n^u$ in Pv. If the approximate factorization of hJ_n is the exact factorization at t_i, $i<n$, then v^* corresponding to the computed $e_{\tau_n}^{v^*}$ will not satisfy $G_n^Tv^* = 0$ exactly, but rather

$$0 = G_i^Tv^* = G_n^Tv^* + hr, \qquad \|r\| = O(1),$$

where $r=-(n-i)(G_n')^T v^*+\cdots$. In sum, we have derived an estimate based on the size of $(e^u_{\tau_n}, e^{v^*}_{\tau_n})$. It is somewhat complicated, but it may be more reliable than basing the estimate solely on τ^u_n.

There are several issues to consider when implementing these alternative error tests (where some variables are excluded from the error tests) in software for the solution of differential/algebraic systems. First, we note that there are two main tests which control the operation of a code—namely, the test to decide when to terminate the Newton iteration, and the error test to accept or reject the current step and to control the stepsize. If we exclude some variables from the error test but base the convergence test on the values of all the variables (or scaled variables), then the weighted max norm seems to be preferable to some other norms. Why? Consider for example the RMS norm, which is used in several popular ODE codes. Since this norm looks at all the variables for the convergence test, but only some of them for the error test, and it is weighted by the number of variables, then something like the following can happen. The code can pass the convergence test very easily, possibly because some very small components are included in the norm for the convergence test, but not really have converged to a great enough accuracy in the variables which are included in the error test. This can cause the code not to run smoothly, or to be unreliable. With the max norm, this kind of incompatibility in norms cannot occur. The second observation is that it is relatively easy and cheap to implement the error test where some variables are excluded in a code which allows the user to write a subroutine to define his own norm. We can pass a flag to the norm routine which tells it whether it wants a norm for the convergence test or for the error test, and then the norm can be computed based on the appropriate variables.

Acknowledgment. We wish to thank Bob Skeel for his comments on an earlier version of this paper, and in particular for suggesting (3.3) as a basis for the roundoff error analysis.

REFERENCES

[1] P. LÖTSTEDT AND L. R. PETZOLD, *Numerical solution of nonlinear differential equations with algebraic constraints* I: *Convergence results for backward differentiation formulas*, Math. Comp., to appear.

[2] K. E. BRENAN, *Stability and convergence of difference approximations for higher index differential-algebraic systems with applications in trajectory control*, Ph.D. Thesis, 1983, Univ. California, Los Angeles.

[3] C. W. GEAR, *Simultaneous numerical solution of differential/algebraic equations*, IEEE Trans. Circuit Theory, CT-18 (1971), pp. 89-95.

[4] J. DIEUDONNÉ, *Foundations of Modern Analysis*, Academic Press, New York, 1969.

[5] P. LÖTSTEDT AND L. R. PETZOLD, *Numerical solution of nonlinear differential equations with algebraic constraints*, SAND83-8877, Sandia National Laboratories, Livermore, CA, 1983.

[6] P. M. GRESHO, R. L. LEE AND R. L. SANI, *On the time-dependent solution of the incompressible Navier-Stokes equations in two and three dimensions*, in Recent Advances in Numerical Methods in Fluids, Pineridge, Swansea, 1980.

[7] J. WITTENBURG, *Dynamics of Systems of Rigid Bodies*, Teubner, Stuttgart, 1977.

[8] C. W. GEAR AND L. R. PETZOLD, ODE *methods for the solution of differential/algebraic systems*, SIAM J. Numer. Anal., 21 (1984), pp. 367-384.

[9] L. PETZOLD, *Differential/algebraic equations are not* ODEs, this Journal, 3 (1982), pp. 367-384.

[10] D. BENSON, personal communication, Lawrence Livermore National Laboratory, Livermore, CA.

[11] N. ORLANDEA, D. A. CALAHAN AND M. A. CHACE, *A sparsity-oriented approach to the dynamic analysis and design of mechanical systems—Part* 1 *and Part* 2, Trans. ASME, J. Engineering for Industry, Ser. B, 99 (1977), pp. 773-784.

[12] J. BAUMGARTE, *Stabilization of constraints and integrals of motion in dynamical systems*, Comput. Meth. Appl. Mech. Engrg., 1 (1972), pp. 1-16.

[13] P. LÖTSTEDT, *On a penalty function method for the simulation of mechanical systems subject to constraints*, TRITA-NA-7919, Royal Institute of Technology, Stockholm, Sweden, 1979.

[14] R. L. SANI, B. E. EATON, P. M. GRESHO, R. L. LE AND S. T. CHAN, *On the solution of the time-dependent incompressible Navier-Stokes equations via a Penalty Galerkin finite element method*, UCRL-85354, Lawrence Livermore National Laboratory, Livermore, CA, 1981.

[15] J. F. PAINTER, *Solving the Navier-Stokes equations with* LSODI *and the method of lines*, Report UCID-19262, Lawrence Livermore National Laboratory, Livermore, CA, 1981.

[16] A. VAN DER SLUIS, *Condition, equilibration and pivoting in linear algebraic systems*, Numer. Math., 15 (1970), pp. 74-86.

[17] L. F. SHAMPINE, *Conditioning of matrices arising in the solution of stiff* ODEs, SAND82-0906, Sandia National Laboratories, Albuquerque, NM, 1982.

[18] L. R. PETZOLD, *A description of* DASSL: *A differential/algebraic system solver*, SAND82-8637, Sandia National Laboratories, Livermore, CA, 1982.

[19] R. L. BROWN AND C. W. GEAR, *Documentation for* DFASUB—*A program for the solution of simultaneous implicit differential and nonlinear equations*, UIUCDCS-R-73-575, Univ. Illinois, Urbana-Champaign, 1973.

[20] R. F. SINCOVEC, B. DEMBART, M. A. EPTON, A. M. ERISMAN, J. W. MANKE AND E. L. YIP, *Solvability of large-scale descriptor systems*, Report, Boeing Computer Services Company, Seattle, WA, 1979.

[21] A. BEN-ISRAEL AND T. N. E. GREVILLE, *Generalized Inverses: Theory and Applications*, John Wiley, New York-London, 1974.