SLAC-PUB-723 (Revised)
August 1970
(MISC)

THE SIMULTANEOUS NUMERICAL SOLUTION OF DIFFERENTIAL — ALGEBRAIC EQUATIONS*

C. W. Gear**

Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

and

Computer Science Department
Stanford University, Stanford, California 94305

(To appear in IEEE Trans. on Circuit Theory.)

---

* Supported in part by the U.S. Atomic Energy Commission.

** On leave from University of Illinois at Urbana, Illinois.

ABSTRACT

This paper discusses a unified method for handling the mixed differential and algebraic equations of the type that commonly occur in the transient analysis of large networks or in continuous system simulation. The first part of the paper is a brief review of existing techniques of handling initial value problems for stiff ordinary differential equations written in the standard form $y' = f(y, t)$. In the second part one of these techniques is applied to the problem $F(y, y', t) = 0$. This may be either a differential or an algebraic equation as $\partial F/\partial y'$ is non-zero or zero. It will represent a mixed system when vectors $\underline{F}$ and $\underline{y}$ represent components of a system. The method lends itself to the use of sparse matrix techniques when the problem is sparse.

I. INTRODUCTION

Many problems in transient network analysis and continuous system simulation lead to systems of ordinary differential equations which require the solution of a simultaneous set of algebraic equations each time that the derivatives are to be evaluated. The "text book" form of a system of ordinary differential equations is

$$\underline{w}' = \underline{f}(\underline{w}, t) \tag{1}$$

where w is a vector of dependent variables, $\underline{f}$ is a vector of functions of $\underline{w}$ and time t, of the same dimension as $\underline{w}$ and $\underline{w}'$ is the time derivative of $\underline{w}$. Most methods discussed in the literature require the equations to be expressed in this form. The "text book" extension to a simultaneous system of differential and algebraic equations (henceforth, D-A E's) could be

$$\begin{aligned} \underline{w}' &= \underline{f}(\underline{w}, \underline{u}, t) \\ 0 &= \underline{g}(\underline{w}, \underline{u}, t) \end{aligned} \tag{2}$$

where $\underline{u}$ is a vector of the same dimension as $\underline{g}$ (but not necessarily the same as $\underline{w}$).

A simple method for initial value problems such as Euler's method has the form

$$\underline{w}_n = \underline{w}_{n-1} + h\,\underline{f}\left(\underline{w}_{n-1}, \underline{u}_{n-1}, t_{n-1}\right) \tag{3}$$

where $h = t_n - t_{n-1}$ is the time increment. Since only $\underline{w}_{n-1}$ is known from the previous time step or the initial values, the algebraic equations

$$0 = \underline{g}\left(\underline{w}_{n-1}, \underline{u}_{n-1}, t_{n-1}\right) \tag{4}$$

must be solved for $\underline{u}_{n-1}$ before each time step.

The properties of the D-A E's typically encountered are:

Differential Equations

Large

Sparse

Stiff

Algebraic Equations

Large

Sparse

Mildly Nonlinear

By sparse we mean that only a few of the variables $\underline{w}$ and $\underline{u}$ appear in any one of the functions $\underline{f}$ or $\underline{g}$. Stiff means that there are greatly differing time constants present, or in other words that the eigenvalues of $\partial \underline{w}'/\partial \underline{w}$ with $\underline{u} = \underline{u}(\underline{w})$ are widely spread over the negative half plane. They are mildly nonlinear in that many of the algebraic variables only appear linearly and that the nonlinearities cause the Jacobians of the system $\partial(\underline{f}, \underline{g})/\partial(\underline{w}, \underline{u})$ to change by only a small amount for small changes in the dependent variable. In regions where the dependent variables are changing rapidly, it is typically necessary to use small integration time steps to follow the solution, so only small changes in the Jacobians occur over each time step.

A further complication that can occur in some problems is that the derivatives $\underline{w}'$ occur implicitly in equations of the form $\underline{F}(\underline{w}, \underline{w}', \underline{u}, t) = 0$. (This complication does not occur in network analysis since techniques have been developed to generate equations in which the derivatives are explicit, although the fact that explicitness is no longer necessary does raise the question of whether the manipulation necessary to put them in this form is worth the effort.)

In order to portray the characteristics of the general problem, we represent it by the set of s equations

$$0 = \underline{H}(\underline{y}, \underline{y}', t) + P\underline{v} \tag{5}$$

rather than the set (2). In Eq. (5) the s dependent variables $\underline{w}$ plus $\underline{u}$ have been replaced by the s variables $\underline{y}$ plus $\underline{v}$. $\underline{v}$ represents the set of s_2 variables that only appear linearly without derivatives. P is an s by s_2 matrix of constants, while $\underline{y}$ is the remaining set of $s_1 = s - s_2$ variables. It is not always convenient to solve (5) for $\underline{y}'$ to get form (2) for several reasons. Among these are:

(i) The $\underline{y}'$ may appear nonlinearly, and hence a numerical solution may require considerable computer time at each step,

(ii) The solution may be ill-conditioned for y' (indeed, for stiff equations it frequently is),

(iii) Writing the equations in form (2) may destroy the sparseness and hence increase solution time.

Suppose that we can partition P by row interchanges into an s_1 by s_2 matrix P_1 and an s_2 by s_2 matrix P_2 which is nonsingular, and that the matching partition of $\underline{H}$ is $\underline{H}_1$ and $\underline{H}_2$ where

$$P = \begin{bmatrix} P_1 \\ \\ P_2 \end{bmatrix} \qquad H = \begin{bmatrix} H_1 \\ \\ H_2 \end{bmatrix}$$

(This is possible if the system (5) has a unique solution.) We can then solve for $\underline{v}$ and write the reduced system as

$$\underline{v} = -P_2^{-1} H_2(\underline{y}, \underline{y}', t) \tag{6}$$

$$0 = H_1(\underline{y}, \underline{y}', t) - P_1 P_2^{-1} H_2(\underline{y}, \underline{y}', t) = \underline{K}(\underline{y}, \underline{y}', t) \tag{7}$$

Equation (7) is a smaller system of D-A E's for $\underline{y}$. However P_2^{-1} is a full matrix in general, so the system is no longer sparse. Since methods that are effective for stiff equations involve the use of the Jacobians of the system, we do not normally want to make this transformation unless s_1 is small.

Our goal is to develop a method that will handle (5) directly, exploit the sparseness, and make use of the linearity of the variables $\underline{v}$.

II. STIFF DIFFERENTIAL EQUATIONS

A natural desire is that the numerical method should be stable whenever the differential system is also stable. Apart from the trapezoidal rule analyzed by Dahlquist[1] little is known about the application of numerical methods to other than the linear system

$$\underline{w}' = A(\underline{w} - \underline{b}(t)) + \underline{b}'(t) = A\underline{w} + \underline{c}(t) \tag{8}$$

where A is a constant square matrix and $\underline{b}(t)$ is a vector function of time. The stability of methods for (8) can be shown to be equivalent to studying the test equations

$$w' = \lambda_i w, \qquad i = 1, 2, \ldots, n \tag{9}$$

where the λ_i are the eigenvalues of the matrix A. Since the solution of (8) is given by

$$\underline{w}(t) = e^{At}(\underline{w}(0) - \underline{b}_0) + \underline{b}(t)$$

we are concerned about restricting the growth of spurious components in the numerical solution compared to the largest growth rate of either $\underline{b}(t)$ or any of the $e^{\lambda_i t}$. Such stability restrictions are dependent on the problem being solved and are hence not useful ones for picking methods suitable for large classes of problems. Dahlquist[1] defined a method to be A-stable if the numerical approximation

w_n it gave to Eq. (9) converged to 0 as n tended to infinity whenever $Re(\lambda_i) < 0$ and the time increment h was fixed and strictly positive. He proved that the maximum order of an A-stable multistep method was 2, and that the A-stable method of order 2 with smallest error coefficient was the trapezoidal rule. For many problems, second order accuracy is not adequate, so we must relax the Dahlquist A-stability criteria if multistep methods are to be used.

For the remainder of this section we will only discuss $w' = f(w, t)$. All comments are applicable to systems. The nature of the problem that arises and the way in which it can be solved is illustrated in Figs. 1 and 2. Figure 1 shows the application of the Euler method $w_n = w_{n-1} + h\, f(w_{n-1}, t_{n-1})$ to the equation

$$w' = \lambda(w - F(t)) + F'(t) \tag{10}$$

where λ is a very negative constant and F(t) is a slowly varying function. Some of the members of the family of solutions are shown in Fig. 1. The Euler method projects the tangent at time t_{n-1} to time t_n. Because λ is so negative, small perturbations from the solutions which tend rapidly to F(t) are amplified rapidly unless h is very small. In fact, the error amplification is a factor of $(1 + h\lambda)$ at each step so that $h\lambda$ must be greater than -2. Figure 2 shows the backward Euler method which is given by

$$w_n = w_{n-1} + h\, f(y_n, t_n) \tag{11}$$

In this method the implicit Eq. (11) must be solved for w_n at each step. It is equivalent to finding the member of the family of solutions for which the tangent at t_n passes through (w_{n-1}, t_{n-1}). The error is amplified by $1/(1 - h\lambda)$ at each step, and so it is stable when $h\lambda$ is in the negative half plane. Any method that is to be stable for arbitrarily negative λ must be implicit, or equivalently, make explicit use of the Jacobian. Methods that are A-stable include the Implicit

Runge Kutta (Ehle),[2] the methods due to Rosenbrock,[3] Calahan[4] and Allen[5] which are extensions of the Runge-Kutta methods that make direct use of the Jacobian, and the methods of the form

$$w_n = w_{n-1} + a_1 h(w'_n + w'_{n-1}) + a_2 h^2(w''_n - w''_{n-1}) + \ldots + a_q h^q\left(w_n^{(q)} \pm w_{n-1}^{(q)}\right) \quad (12)$$

which are of order 2q. The application of these methods is a major computing task for large systems and is not generally practical.

An alternative to requiring A-stability was proposed in Gear.[6] It was suggested that stability was not necessary for values of $h\lambda$ close to the imaginary axis but not close to the origin. These correspond to oscillating components that will continue to be excited in nonlinear problems. It will therefore be necessary to reduce h such that successive values of t_n occur at least about 10 times per cycle. Methods that were stable for all values of $h\lambda$ to the left of $\mathrm{Re}(h\lambda) = -D$ where D was some positive constant, and accurate close to the origin as shown in Fig. 3 were said to be stiffly stable. The multistep methods of order k given by

$$w_n = \alpha_1 w_{n-1} + \ldots + \alpha_k w_{n-k} + h\beta_0 w'_n \quad (13)$$

were shown to be stiffly stable for $k \leq 6$ in Ref. 6. The α_i and β_0 are uniquely determined by the requirement that the order be k.

Unfortunately Eq. (13) is an implicit equation for w_n since $w'_n = f(w_n, t_n)$. In the case that $\partial f/\partial w$ is large the conventional corrector iteration will not converge. However a Newton iteration can be used. If the first iterate is obtained from an explicit predictor formula, few iterations are needed. Furthermore, the first iterate can be made arbitrarily accurate by choosing h small enough. Since Newton's method is locally convergent, we are guaranteed that we will get convergence for small enough h.

In Ref. 7 conventional predictor corrector schemes are re-expressed in a matrix notation that will allow us to equate the methods for differential and algebraic equations. We can develop a similar formulation when Newton's method is used as follows: Let the predicted value of w_n be $\tilde{w}_n$, and suppose it is obtained from the formula

$$\tilde{w}_n = \tilde{\alpha}_1 w_{n-1} + \ldots + \tilde{\alpha}_k w_{n-k} + \tilde{\beta}_1 w'_{n-1}$$

Subtracting this from (13) we get

$$w_n = \tilde{w}_n + \beta_0 \left[h\, f(w_n, t_n) - (\gamma_1 w_{n-1} \cdots + \gamma_k w_{n-k} + \delta_1 h w'_{n-1}) \right]$$

where

$$\gamma_i = (\tilde{\alpha}_i - \alpha_i)/\beta_0$$

and

$$\delta_1 = \tilde{\beta}_1/\beta_0$$

Let us use the symbol $h\tilde{w}'_n$ for $\gamma_1 w_{n-1} + \ldots + \gamma_k w_{n-k} + \delta_1 h w'_{n-1}$. Thus we have

$$w_n = \tilde{w}_n + \beta_0 \left[h\, f(w_n, t_n) - h\tilde{w}'_n \right]$$

and, trivially,

$$h w'_n = h\tilde{w}'_n + \left[h\, f(w_n, t_n) - h\tilde{w}'_n \right]$$

Define the vector $\underline{\omega}_n$ to be $\left[w_n, h w'_n, w_{n-1}, \ldots, w_{n-k+1} \right]^T$ where T is the transpose operator, and $\underline{\tilde{\omega}}_n$ to be $\left[\tilde{w}_n, h\tilde{w}'_n, w_{n-1}, \ldots, w_{n-k+1} \right]^T$. Then we can write

$$\underline{\tilde{\omega}}_n = B \underline{\omega}_{n-1} \tag{14}$$

$$\underline{\omega}_n = \underline{\tilde{\omega}}_n + \underline{c}\, b \tag{15}$$

where

$$\underline{c} = \left[\beta_0, 1, 0, \ldots, 0\right]^T$$

$$B = \begin{bmatrix} \tilde{\alpha}_1 & \tilde{\beta}_1 & \tilde{\alpha}_2 & \cdots & \tilde{\alpha}_{k-1} & \tilde{\alpha}_k \\ \gamma_1 & 1 & \gamma_2 & \cdots & \gamma_{k-1} & \gamma_k \\ 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \cdot & \cdot & \cdot & & & \\ 0 & & & \ldots & 1 & 0 \end{bmatrix}$$

and $b = \left[hf(w_n, t_n) - h\tilde{w}'_n\right]$ is a scalar such that (15) satisfies $hf(w_n, t_n) - hw' = 0$. (Here hw'_n is the second component of $\underline{\omega}_n$ given by (15).) We can subject (14) and (15) to nonsingular linear transformations. In particular, if we note that the components of $\underline{\omega}_n$ represent a unique polynomial $W_n(t)$ of degree k such that $W_n(t_{n-i}) = w_{n-i}$, $0 \le i < k$, and $W'_n(t_n) = w'_n$, we can find the transformation which gives the components of $\underline{a}_n = \left[w_n, hw'_n, h^2w''_n/2, \ldots, h^k w_n^{(k)}/k!\right]^T$ representing the same polynomial. If this transformation is Q we have

$$\tilde{\underline{a}}_n = Q\tilde{\underline{\omega}}_n = Q\,B\,Q^{-1}\,\underline{a}_{n-1} = A\,\underline{a}_{n-1}$$

$$\underline{a}_n = Q\underline{\omega}_n = Q\tilde{\underline{\omega}}_n + Q\,\underline{c}\,b = \tilde{\underline{a}}_n + \underline{\ell}\,b \qquad (16)$$

where $A = Q\,B\,Q^{-1}$ is shown to be the Pascal triangle matrix in Ref. 7, and the values of $\underline{\ell} = Q\,\underline{c}$ are given in Ref. 6 for stiff methods of various orders.

We can determine b from

$$\begin{aligned} 0 &= h\,f(w_n, t_n) - hw'_n \\ &= F(\underline{a}_n, t_n) \quad \text{(by definition)} \\ &= F(\tilde{\underline{a}}_n + \underline{\ell}\,b,\ t_n) \end{aligned} \qquad (17)$$

If this is solved by a Newton iteration starting from $b_{(0)} = 0$, we get

$$b_{(m+1)} = b_{(m)} - \left[d\tilde{F}(b_{(m)})/db\right]^{-1} \tilde{F}(b_{(m)}) \tag{18}$$

where

$$\tilde{F}(b) = F(\tilde{a}_n + \underline{\ell}\, b,\ t_n)$$

Writing

$$\underline{a}_{n,(m)} = \tilde{a}_n + \underline{\ell}\, b_{(m)}$$

we get

$$\underline{a}_{n,(0)} = A\, \underline{a}_{n-1}$$

$$\underline{a}_{n,(m+1)} = \underline{a}_{n,(m)} - \underline{\ell}\left[\sum_{i=0}^{k} \frac{\partial F}{\partial a_i}\, \ell_i\right]^{-1} F(\underline{a}_{n,(m)}, t_n) \tag{19}$$

where a_i is the i^{th} component of $\underline{a}$, numbering from 0. We now note that this technique is not dependent on the differential equations being written in the explicit form (1). It can be applied directly to $F(\underline{a}, t) = K(w, w', t) = 0$ provided that approximations to the Jacobians can be found. Errors in these approximations do not affect the error in the numerical solution, they only effect the rate of convergence of (19).

III. ALGEBRAIC EQUATIONS

If Eqs. (2) are to be solved by the implicit algorithm (13), we must solve the simultaneous algebraic system

$$\underline{w}_n = \alpha_1 \underline{w}_{n-1} + \ldots + \alpha_k \underline{w}_{n-k} + h\,\beta_0\, f(\underline{w}_n, \underline{u}_n, t_n)$$

$$0 = \underline{g}(\underline{w}_n, \underline{u}_n, t_n)$$

for $\underline{w}_n$ and $\underline{u}_n$ at each step. Calahan[9] and Hachtel et al.,[10] do this by a simultaneous Newton iteration. The interesting and useful point to be brought out in this section is that it is not necessary to distinguish between the algebraic and differential variables, so that system (5) can be handled directly.

For the remainder of this section we discuss the single equation $g(u, t) = 0$. All comments apply to systems. If the values of $u(t)$ are known at a number of previous time steps t_{n-i}, $1 \le i \le k+1$ we can approximate $\tilde{u}_n = u(t_n)$ by extrapolation, that is by evaluating the unique k^{th} degree polynomial that passes through these u_{n-i}. Calling this approximation u_n we have

$$\tilde{u}_n = \eta_1 u_{n-1} + \ldots + \eta_{k+1} \; u_{n-k+1}^{\;T}$$

We write

$$\underline{\mu}_n = \left[u_n, u_{n-1}, \ldots, u_{n-k+1}\right]^T \text{ and } \underline{\tilde{\mu}}_n = \left[\tilde{u}_n, \ldots, u_{n-k+1}\right]^T$$

We then have

$$\underline{\tilde{\mu}}_n = E \, \underline{\mu}_{n-1} \tag{20}$$

where

$$E = \begin{bmatrix} \eta_1 & \eta_2 & \cdots & \eta_k & \eta_{k+1} \\ 1 & 0 & & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ \cdots & & \cdots & & \cdots \\ 0 & 0 & & 1 & 0 \end{bmatrix}$$

Let b be such that u_n, the solution of $g(u_n, t_n) = 0$, satisfies

$$u_n = \tilde{u}_n + b$$

If we define $\underline{e} = \left[1, 0, \ldots, 0\right]^T$, we can write

$$\underline{\mu}_n = \underline{\tilde{\mu}}_n + \underline{e} \, b \tag{21}$$

b is such that $\underline{\mu}_n$ satisfies

$$\begin{aligned} g(u_n, t_n) &= G(\underline{\mu}_n, t_n) \quad \text{(by definition)} \\ &= G(\underline{\tilde{\mu}}_n + \underline{e} \, b, \; t_n) = 0 \end{aligned} \tag{22}$$

We can represent the k^{th} degree polynomial that was used to extrapolate from u_{n-i} to $\tilde{u}_n$ by the values of its derivatives at t_{n-1}, that is by the components of

$\underline{a}_{n-1} = \left[u_{n-1}, hu'_{n-1}, \ldots, h^k u^{(k)}_{n-1}/k!\right]^T$. When we apply the appropriate linear transformation S to obtain $\underline{a}_{n-1} = S\,\underline{\mu}_{n-1}$ we have from (20)

$$\underline{\tilde{a}}_n = S\,\underline{\tilde{\mu}}_n = S\,E\,S^{-1}\,\underline{a}_{n-1}$$

and from (21)

$$\underline{a}_n = \underline{\tilde{a}}_n + S\,\underline{e}\,b \tag{23}$$

The matrix $S\,E\,S^{-1}$ is the pascal triangle matrix A and the vector $S\,\underline{e}$ is identical to the vector $\underline{\ell}$ (see Ref. 11) so these equations are identical to Eq. (16) derived for differential equations. Therefore, iteration (19) can be used to solve Eqs. (22) and (23) for b if we identify G with F. (They are both functions of t and the first component of $\underline{a}$ only, namely u_n or w_n respectively.)

IV. MIXED SYSTEMS

The equation $F(\underline{a}, t) = K(w, w', t) = 0$ is a differential equation if $\partial K/\partial w' \neq 0$, otherwise it is an algebraic equation. Because the above techniques for differential and algebraic equations are identical, it is not necessary to distinguish in the cases. The fact that the Jacobian $\partial K/\partial w' = h\partial F/\partial a_1$ appears in (19) causes the method to adjust appropriately. If the system (7) is to be solved, we handle it as follows. Let the i^{th} components of $\underline{y}$ and $\underline{y}'$ be carried along with higher order derivatives as $\underline{a}^i$, $1 \leq i \leq s$. We write the j^{th} equation from (7) as

$$F^j(\{\underline{a}^i\}, t) = 0$$

The prediction step is

$$\underline{a}^i_n = A\,\underline{a}^i_{n-1}$$

and the Newton corrector iteration (19) is

$$\sum_{i=1}^{s}\left[\sum_{q=0}^{k}\frac{\partial F^j}{\partial a_q^i}\,\ell_q\right]\left(\underline{a}^i_{n,(m)} - \underline{a}^i_{n,(m+1)}\right) = \underline{\ell}\,F^j\left(\left\{\underline{a}^i_{n,(m)}\right\},\, t_n\right) \qquad (24)$$

We know that by choosing sufficiently small h, the predictor can be made arbitrarily accurate for both algebraic and differential variables, so, for small enough h, we will be within the region of convergence of Newton's method.

The linear system (24) must be solved for each correction iteration. If the Jacobians are sparse, appropriate techniques can be used (see Hachtel et al.,[10]). If, as is common, they are slowly varying, they need not be re-evaluated at each iteration, although Calahan[9] suggests that it is valuable to evaluate them immediately after each prediction step.

It is possible to avoid the work of predicting the first approximation of the linear variables $\underline{v}$ in (5). Initial errors in $\underline{v}_{n,(0)}$ do not affect the corrector iteration for the nonlinear algebraic or differential equations. This can be seen by considering the partition used to get (6) and (7). Iteration (24) will take the form

$$\begin{bmatrix} \frac{\partial \underline{H}_1}{\partial \underline{y}}\,\ell_0 + \frac{\partial \underline{H}_1}{\partial \underline{y}'}\,\frac{\ell_1}{h} & P_1\ell_0 \\ \frac{\partial \underline{H}_2}{\partial \underline{y}}\,\ell_0 + \frac{\partial \underline{H}_2}{\partial \underline{y}'}\,\frac{\ell_1}{h} & P_2\ell_0 \end{bmatrix} \begin{bmatrix} \Delta\underline{y} \\ \Delta\underline{v} \end{bmatrix} = -\ell_0 \begin{bmatrix} \underline{H}_1(\underline{y},\underline{y}',t) + P_1\underline{v} \\ \underline{H}_2(\underline{y},\underline{y}',t) + P_2\underline{v} \end{bmatrix} \qquad (25)$$

where $\Delta\underline{y}_n = \underline{y}_{n,(m+1)} - \underline{y}_{n,(m)}$ and $\Delta\underline{v}$ is similar. Premultiply this by

$$\begin{bmatrix} I & P_1P_2^{-1} \\ 0 & I \end{bmatrix}$$

to get

$$\begin{bmatrix} \left(\ell_0 \frac{\partial}{\partial \underline{y}} + \frac{\ell_1}{h} \frac{\partial}{\partial \underline{y}'}\right)\left(\underline{H}_1 - P_1 P_2^{-1} \underline{H}_2\right) & 0 \\ \left(\ell_0 \frac{\partial}{\partial \underline{y}} + \frac{\ell_1}{h} \frac{\partial}{\partial \underline{y}'}\right) & \underline{H}_2 \qquad P_2 \ell_0 \end{bmatrix} \begin{bmatrix} \Delta \underline{y} \\ \Delta \underline{v} \end{bmatrix} = -\ell_0 \begin{bmatrix} \underline{H}_1 - P_1 P_2 \underline{H}_2^{-1} \\ \underline{H}_2 \end{bmatrix}$$

The first rows of this iteration are those that would arise if (7) were solved by the same technique, and are independent of $\underline{v}$. Therefore the errors in $\underline{v}$ do not affect the iteration, so only the current value $\underline{v}_n$ need be saved from step n. "Prediction" can consist of the step $\underline{v}_{n,(0)} = \underline{v}_{n-1}$ which requires no computation for the linear variables.

V. PROGRAM ORGANIZATION

In this section we give an outline of the organization of a subroutine which generates the values of the vectors $\underline{a}_n^i$ from the values of $\underline{a}_{n-1}^i$. (This subroutine is available from the author, although it is currently "experimental," that is, inadequately documented. It is an extension of the subroutine for stiff ordinary differential equations given in Ref. 8.)

The subroutine calls on a lower level user supplied subroutine which must evaluate the components of $\underline{H}(\underline{y}, \underline{y}', t) + P\underline{v}$ when values of $\underline{y}$, $\underline{y}'$, $\underline{v}$ and t are given. It chooses the order of the method and the step size automatically. Starting is accomplished by setting the order of the method to one so that the only components of $\underline{a}_0$ are y_0 which is given and hy'_0 which is set to zero. The program flow is outlined in Fig. 4. The main inner loop is a correction loop which may be traversed up to three times. If corrections are not small by then the Jacobian is re-evaluated. If this has already been tried, the step size is reduced by a factor of 4 to try and get into the region of convergence for Newton's method. When

small corrections are obtained, the differences between the predictor and corrector are used as estimates of the single step truncation error. If this is larger than a user supplied parameter, the step is reduced by an amount dependent on the order, and the step is repeated. If the error is acceptable, the step sizes that could be used for this order, one order higher, and one order lower are estimated. The largest of these is chosen for the next step, and the appropriate order is selected. Details of this and many other minor problems that must be handled in a practical integration algorithm are similar to those in the package given in Ref. 8 for differential equations, and described in Ref. 12.

VI. A NUMERICAL EXAMPLE

The following example is "artificial" in that it is not derived from a network. Rather it consists of a set of stiff differential equations coupled to a set of differential and algebraic equations. These have been chosen because enough of the solution is known to permit checking of the answers, and because they illustrate the features of the method.

A system of four differential equations has been proposed by Krogh (private communication) to test stiff equations. They are given by:

$$0 = y_i' - s + (r - y_i)^2 + \sum_{j=1}^{4} b_{ij}\, y_j \qquad \text{for } i = 1 \text{ to } 4 \tag{26}$$

where

$$r = (y_1 + y_2 + y_3 + y_4)/2$$

$$s = \sum_{i=1}^{4} (r - y_i)^2/2$$

and b_{ij} is a symmetric matrix with

$$b_{11} = b_{22} = b_{33} = b_{44} = 447.50025$$

$$b_{12} = -b_{34} = b_{21} = -b_{43} = -452.49975$$

$$b_{13} = -b_{24} = b_{31} = -b_{42} = -47.49975$$

$$b_{14} = -b_{23} = b_{41} = -b_{32} = -52.50025$$

Their solution is given by $y_i = p - z_i$ where

$$p = (z_1 + z_2 + z_3 + z_4)/2$$

and

$$z_i = \beta_i / \left(1 + c_i e^{\beta_i t}\right)$$

(This is the solution of $z_i' = z_i^2 - \beta_i z_i$.) The starting conditons are $y_i = -1$, so that $c_i = -(1 + \beta_i)$. (The values of β_i used are $\beta_1 = 1000$, $\beta_2 = 800$, $\beta_3 = -10$, and $\beta_4 = 0.001$.) As t tends to infinity the eigenvalues of the system approach $-|\beta_i|$. To this system we add the four equations

$$0 = y_5' + y_1 y_6' + y_1' y_6$$

$$0 = 2y_6 + y_6^3 - y_1 + v_1 - 1 - e^{-t} = F_6$$

$$0 = v_1 - v_2 + y_1 y_6 = F_7$$

$$0 = v_1 + v_2 + 5y_1 y_2 = F_8 \qquad (27)$$

with the initial conditions $y_5 = y_6 = 1$, $v_1 = -2$, and $v_2 = -3$. It is not immediately evident whether these represent differential equations for y_5, y_6, or both! In fact they were chosen so that the first of these equations can be integrated to give

$$0 = y_5 + y_1 y_6 = F_5 \qquad (28)$$

The system of 8 Eqs. (26) and (27) have been integrated by the technique described. The maximum error in y_1 to y_4 and the residuals F_5 to F_8 are shown in Table I

at times t = .01 and t = 1000, along with the number of steps, evaluations of equations, and evaluations of the Jacobian and subsequent matrix inversion. The single step error control parameter was varied from 10^{-4} to 10^{-8}. The integration was started at t = 0. By t = .01 the most rapidly changing component was down by a factor of about e^{-10}, by t = 1000, the system was approaching equilibrium. (The slowest component was causing changes in the fourth significant digit.) A conventional integration scheme would fail because of the stiffness of the differential equations. (Adam's method takes from 3 to 10 million equation evaluations for this problem depending on the error control parameter.)

VII. SUMMARY

If the mathematical model has a unique solution, the numerical method presented will compute it, although for badly conditioned problems the step size may be small. Even if the algebraic equations permit more than one solution, the method will follow the one smoothly connected to the initial values as long as the multiple solutions remain distinct to within the error criteria. In some cases the method will extrapolate through a multiple root of the algebraic equations providing that the solution is sufficiently smooth, but usually, near a multiple root, the corrector iteration will fail to converge although the step size will have been reduced to the minimum allowed. (This is a user parameter to the program.)

A current problem, common to other methods, is that the initial values of the variables $\underline{y}$ and $\underline{v}$ must be provided. This means that a partial solution of the algebraic equations is necessary initially. Currently an investigation of techniques for solving algebraic equations by means of differential equations is underway. It is expected that this will make it possible to use the same program to generate those initial conditions not supplied by the user.

The method is similar to that used in Refs. 9 and 10, in fact the method for differential equations is identical. However this method has the advantage that the same technique is applied to both algebraic and differential variables, making it unnecessary to distinguish between them. This has important consequences for continuous system simulation, and can have consequences for network analysis since it will no longer be necessary to manipulate the equations into a "text book" form.

REFERENCES AND FOOTNOTES

1. G. Dahlquist, B.I.T. 3, 27-43 (1967).
2. B. L. Ehle, B.I.T. 8, 276-278 (1968).
3. H. H. Rosenbrock, Comp. J. 5, 329 (1963).
4. D. Calahan, "A stable accurate method of numerical integration for nonlinear systems," IEEE Proceedings, p. 744 (1968).
5. R. H. Allen, "Numerically stable explicit integration techniques using a linearized Runge-Kutta extension," Boeing Scientific Laboratories Document d1-82-0929 (1969).
6. C. W. Gear, "The automatic integration of stiff ordinary differential equations," Information Processing 68. Ed. Morrell, (North Holland Publishing Co., Amsterdam, 1969); p. 187-193.
7. C. W. Gear, Math. Comp. 21, 146-156 (1967).
8. C. W. Gear, "An algorithm for the integration of ordinary differential equations," (to appear in C.A.C.M., in the Algorithms section).
9. D. Calahan, "Numerical considerations in the transient analysis and optimal design of nonlinear circuits," Digest Record of Joint Conference on Mathematical and Computer Aids to Design, ACM/SIAM/IEEE, (1969).
10. Hachtel, Brayton and Gustavson, "The sparse tableux approach to network analysis and design," this issue.

11. That the matrices $Q\ B\ Q^{-1}$ and $S\ E\ S^{-1}$ are both the Pascal triangle matrix A follows from the fact that B and E represent the highest possible order (k) extrapolation process possible in their respective representations, as does A. They must, therefore, be equivalent.

    For small ϵ the differential equation $\epsilon w' = f(w, t)$ is stiff. The limit of this as ϵ tends to 0 is the algebraic equation $0 = f(w, t)$. If we take the limit of the

numerical method (13) as ϵ tends to 0 for this problem we find ourselves solving $h\beta_0 f(w_n, t_n) = 0$. Thus the limit of iteration (19) as ϵ tends to 0 will give a technique for solving algebraic equations. Its matrix will be A and its vector $\underline{\ell}$. The error amplification matrix for both problems (the relation between the errors in $\underline{a}_n$ and errors in $\underline{a}_{n-1}$) will have all zero eigenvalues since the error at t_n is independent of earlier errors when the corrector is iterated to convergence for an algebraic equation. In Ref. 7 it is shown that the eigenvalues uniquely determine the vector $\underline{\ell}$ for this problem, hence $\underline{\ell}$ and S $\underline{e}$ are identical.

12. C. W. Gear, "The automatic integration of ordinary differential equations," (to appear in C.A.C.M.)

TABLE I

Error Control Parameter	Max. Error in y_1 to y_4	$\|F_5\|$	$\|F_6\|$	$\|F_7\|$	$\|F_8\|$	No. Steps	No. Equation Evaluations	No. Jacobian Evaluations
t = .01								
1.0 - 4*	3.8 - 5	1.2 - 7	2.2 - 9	5.4 - 10	3.2 - 13	41	197	12
1.0 - 5	4.8 - 7	1.8 - 6	7.5 - 10	4.4 - 11	2.5 - 10	55	235	13
1.0 - 6	2.7 - 7	3.3 - 7	3.4 - 11	7.8 - 12	6.7 - 15	70	270	13
1.0 - 7	6.2 - 8	2.7 - 7	7.3 - 13	0	2.2 - 16	85	377	18
1.0 - 8	1.7 - 9	2.8 - 8	2.0 - 12	5.0 - 13	2.2 - 16	137	608	27
t = 1000.								
1.0 - 4	1.0 - 5	3.3 - 3	4.9 - 10	4.5 - 12	4.2 - 11	168	937	54
1.0 - 5	1.1 - 5	4.7 - 4	1.6 - 11	2.1 - 12	2.0 - 11	214	1059	53
1.0 - 6	1.1 - 6	2.0 - 5	3.8 - 13	1.2 - 13	2.1 - 13	305	1389	61
1.0 - 7	4.7 - 7	1.4 - 6	3.6 - 15	0	3.6 - 15	379	1779	75
1.0 - 8	3.8 - 8	5.3 - 5	0	3.6 - 15	7.1 - 15	530	2894	138

*This notation means 1.0×10^{-4}.

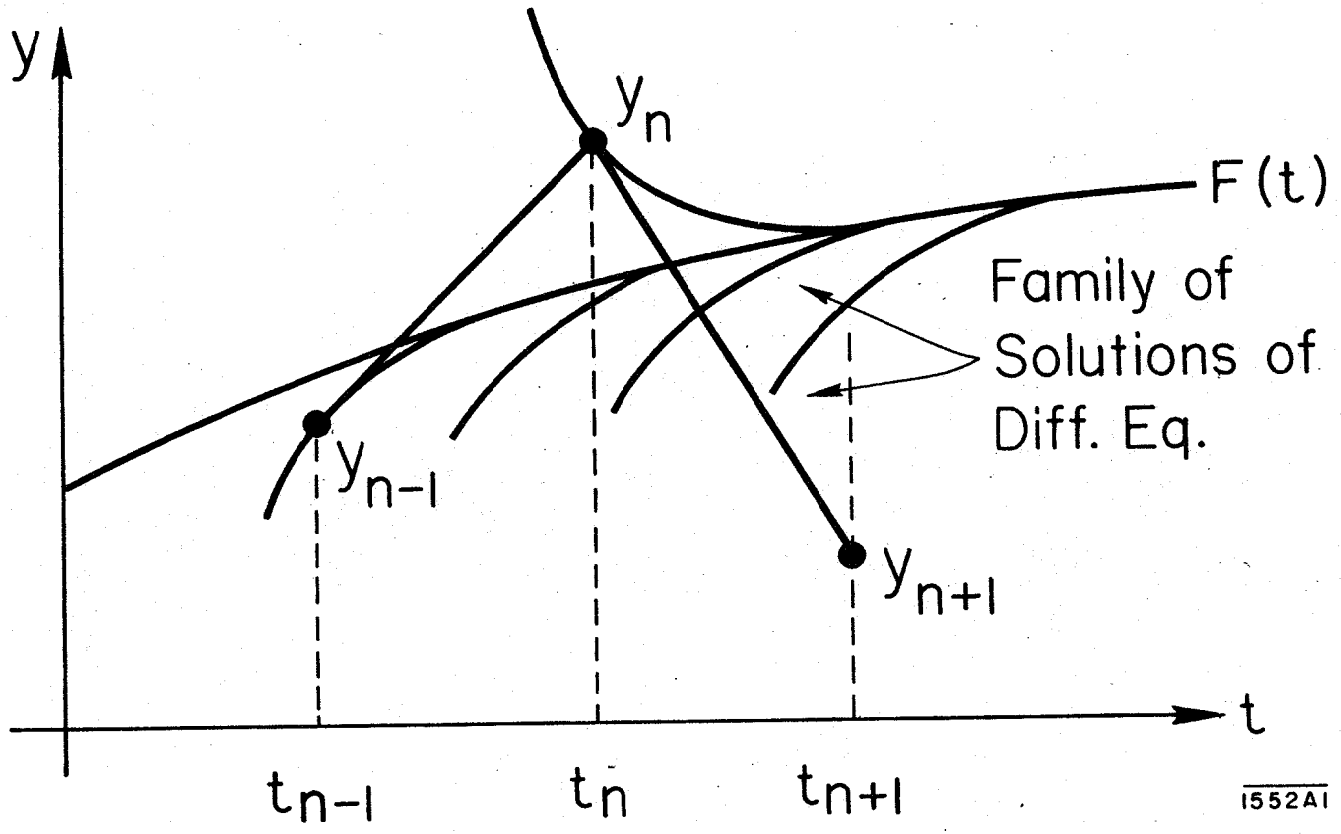

Fig. 1

Euler solution of (10).

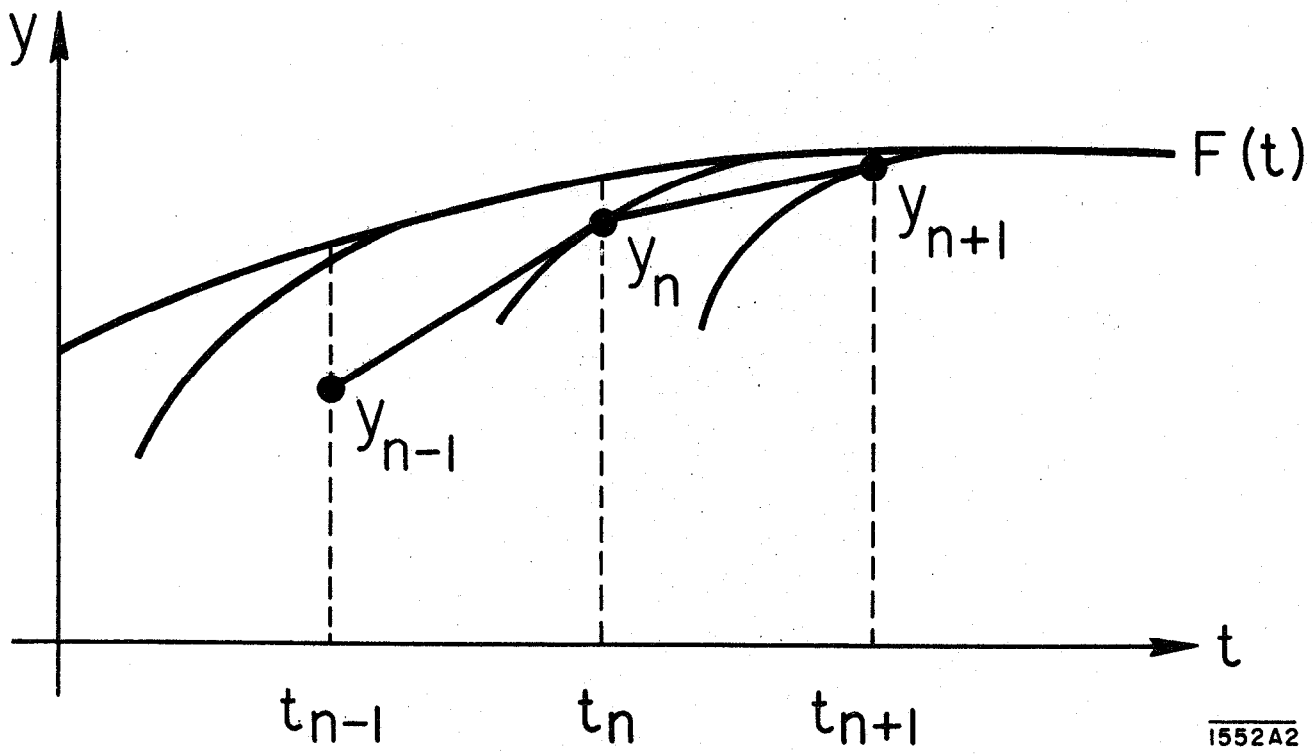

Fig. 2

Backward Euler solution of (10).

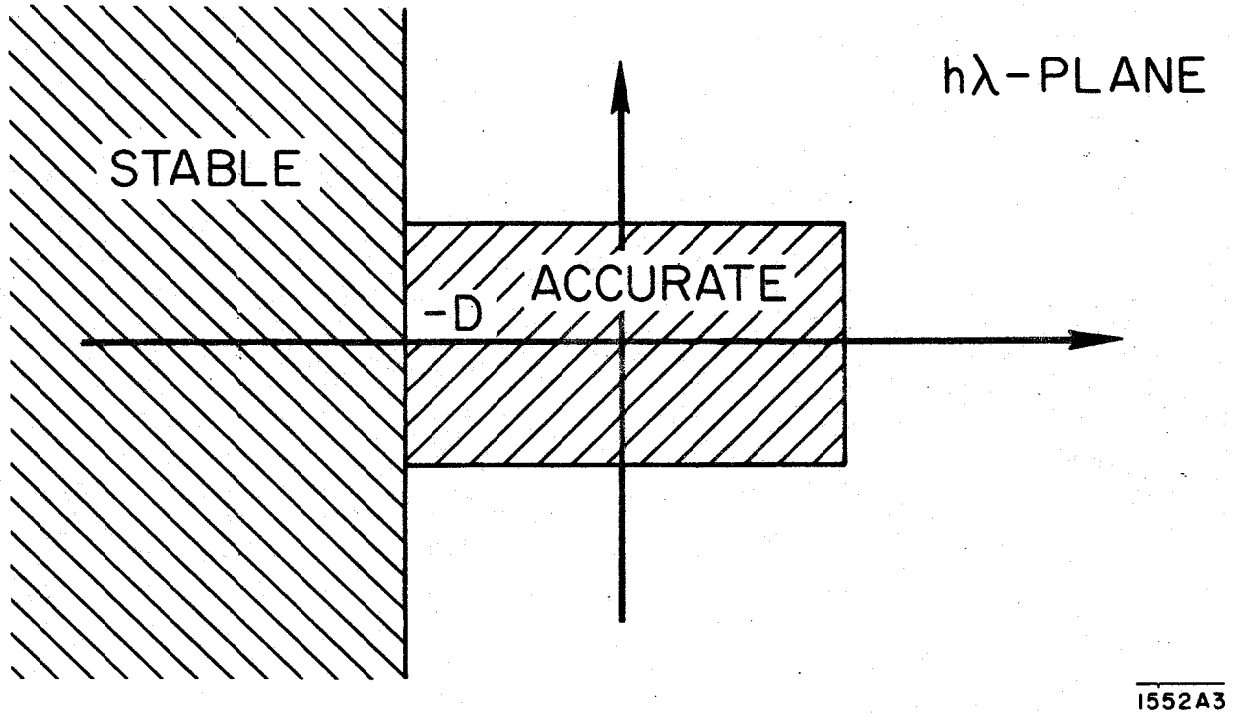

Fig. 3

Stability regions.

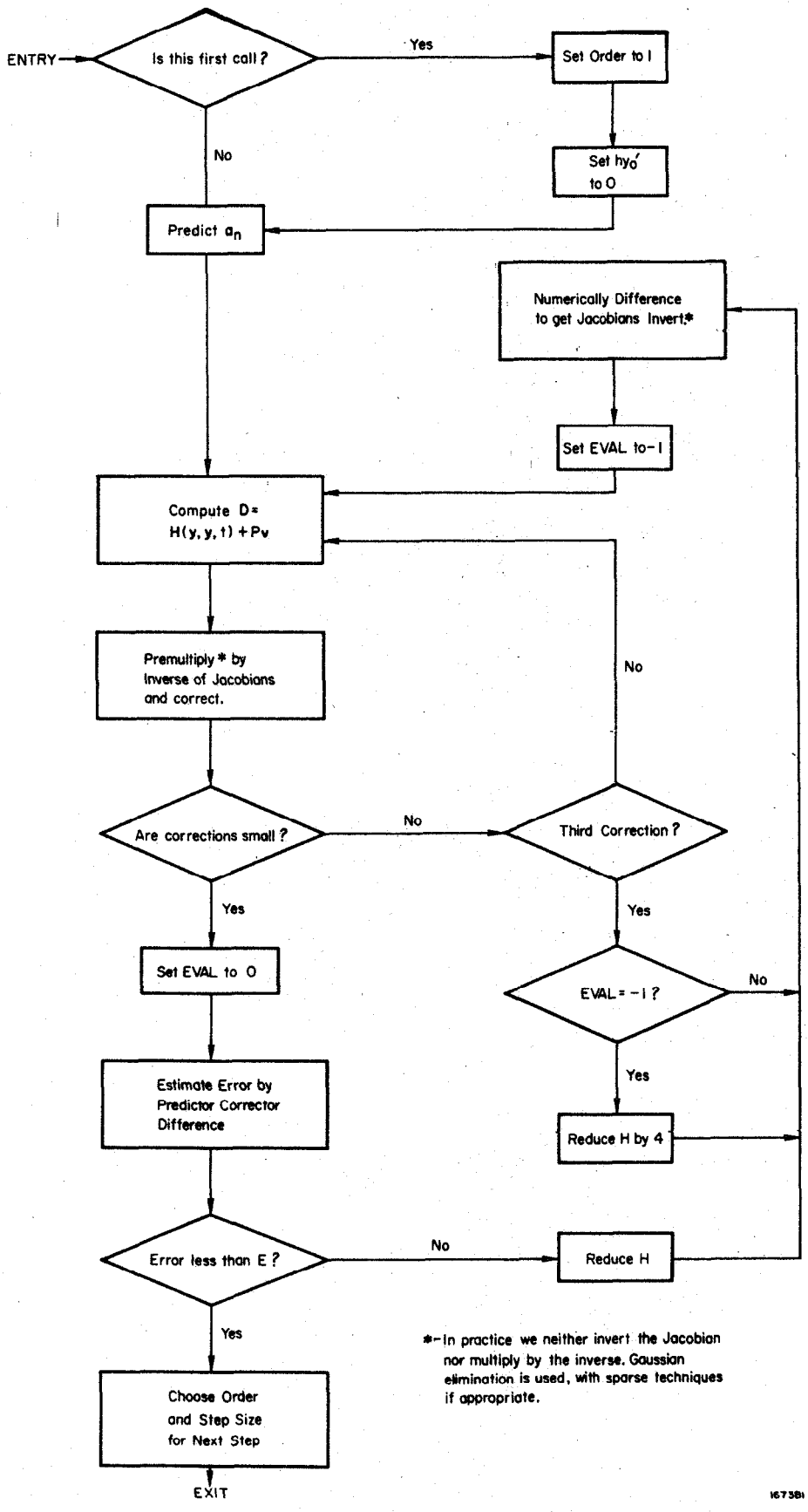

Fig. 4

Program organization.