Foundations of Computer Aided Process Design, Mah & Seider (eds.) Engineering Foundation (1981)

Numerical Solution of Differential Equations—An Overview

Brice Carnahan, James O. Wilkes
University of Michigan, Ann Arbor, MI 48109

ABSTRACT

Numerical solution techniques for ordinary (ODE) and partial (PDE) differential equations are described with emphasis on methods that have been, are being, or seem likely to be used in the solution of complex simulation problems. Specialized algorithms are avoided in favor of those that might reasonably be incorporated into general-purpose software, suitable for solving essentially arbitrary systems of ODE or PDE.

Four major topics are covered: (1) Initial value (IV) methods for ODE, (2) Boundary value (BV) methods for ODE, (3) Finite-difference methods (FDM) for PDE, and (4) Finite-element methods (FEM) for PDE.

Classical Runge-Kutta (RK) and linear-multistep (LMS) methods for the solution of the IV ODE problem are discussed, but principal emphasis in the first section is on numerical stability and its relationship to the stiff equation problem. LMS and semi-implicit RK methods suitable for stiff equations are included. The section ends with comments on accuracy and stepsize and order-changing strategies.

The most important of the BV ODE methods, including shooting,

parallel shooting, finite- difference approximation, quasilinearization, and collocation are covered in the second section. These methods are less reliable and more costly than the IV ODE algorithms.

The third section starts with a classification of PDE as to type (parabolic, elliptic, hyperbolic) and various finite difference approximations. FDM methods for parabolic equations in one space dimension are traced from simple explicit, through Crank-Nicolson, DuFort-Frankel, and Saul'yev methods with stability analyses. The implicit-alternating direction method is shown for two and three space dimensions. The method of characteristics is recommended for hyperbolic systems. Elliptic systems lead to large systems of simultaneous equations for which several solution techniques are possible. The method of lines is introduced as a broad tool for solving PDE's of all types.

In the last section, the FEM method is developed for solution of PDE such as Lu = f subject to essential, natural, and mixed boundary conditions. The solution is approximated by trial functions and Galerkin's principle is invoked as the basic tool for determining nodal solution estimates. A Green's formula is applied to reduce the order of the highest-order derivatives. Introduction of finite elements, such as triangles, then enables the basis functions to be specified and reduces the problem to that of solving a system of linear algebraic equations for the estimates of the nodal solution values. The paper concludes with the FEM solution (in space) of a time-dependent (parabolic) PDE, which leads to the solution of simultaneous ODE in the nodal solution values.

SCOPE

Engineering problems involving spatially-distributed and/or time-dependent variables invariably lead to mathematical models containing ordinary and partial differential equations. The non-linearities, irregular geometries, and/or the nature of boundary conditions usually present in realistic models preclude the finding of analytical solutions. We must generate instead admittedly less general and approximate numerical methods.

We give an overview of numerical solution techniques for ordinary (ODE) and partial (PDE) differential equations. Complete coverage of so broad an area is obviously not possible within the context of a conference presentation. We concentrate on those numerical methods that have been, are being, or seem likely to be used extensively in the solution of complex simulation problems. We avoid rather specialized methods (for example, methods for ODE of order greater than one) in favor of those that might reasonably be incorporated into general-purpose software packages. Additionally, we assume that most conference attendees have had considerable computing experience, and have probably already solved some differential equations by classical techniques. Thus, explicit Runge-Kutta methods and the better known of the linear multistep methods are given lighter coverage than are numerical stability, stiff systems of ODE, and methods suitable for solving stiff problems.

We have not attempted to prepare a comprehensive review of the latest research papers on solution methods for ODE and PDE. The literature abounds with new methods (particularly for solution of the IV ODE problem), often not compared carefully (or fairly) with existing

methods, and usually not tested on practical problems. [Admittedly, this is very difficult to do in a comprehensive way, requiring the use of many different routines of varying quality and robustness.] New methods usually achieve acceptance only after extensive testing by users (often only when incorporated into generally available software). Hence, our presentation of what are accepted approaches or new techniques that appear particularly promising will have more the appearance of a textbook (in the style of our text, Applied Numerical Methods (19)) than of a traditional review paper.

The paper is divided into four major sections: (1) Initial value methods for ODE, (2) Boundary value methods for ODE, (3) Finite-difference methods for PDE, and (4) Finite-element methods for PDE.

Some important topics (in particular, description of the many excellent software packages available for solution of ODE and PDE, collocation, and a detailed treatment of the method of lines solution of PDE) are not included, as they will be covered in papers by Byrne, Michelsen and Carver, respectively, also appearing in these proceedings.

INITIAL-VALUE METHODS FOR THE SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

INTRODUCTION

Many steady-state processes with spatially distributed parameters and unsteady-state processes with lumped parameters can be modeled with one or more ordinary differential equations (ODE) of the form

$$F(x, y, y', y'', \ldots, \frac{d^n y}{dx^n}) = 0 \qquad (1.1)$$

normally subject to n conditions (values of y and/or of its various derivatives at specified values of x). If all conditions are specified at a common point, x_0, then the problem is termed an initial value (IV) problem; if not, the problem is called a boundary value (BV) problem. Methods for the initial value problem are well developed and reliable; those for the boundary value problem are, in general, less satisfactory and computationally more complex and expensive. In this section we shall treat the IV problem; BV solution methods will be discussed in the next section.

Since nth-order equations of the form of (1.1) can always be rewritten as a system of n first-order ODE by defining n-1 new dependent variables, we need only consider the solution of systems of first-order equations. Many special methods, particularly for second-order ODE, are available and when applicable are sometimes computationally more efficient than equivalent first-order methods. Nevertheless, for general-purpose software, only systems of first-order equations of the form

$$\frac{dy_j}{dx} = y'_j(x) = f_j(x, y_1, y_2, \ldots, y_n), \qquad j=1,2,\ldots,n, \qquad (1.2)$$

$$y_j(x_0) = y_{j0},$$

are usually considered; equations (1.2) are often written in vector

form

$$\frac{d\bar{y}}{dx} = \bar{y}'(x) = \bar{f}(x,y)\,, \qquad \bar{y}(x_0) = \bar{y}_0\,, \tag{1.3}$$

or in autonomous vector form

$$\bar{z}' = \bar{F}(\bar{z})\,, \qquad \bar{z}(x_0) = \bar{z}_0\,, \tag{1.4}$$

where $\bar{z} = (\bar{y}\ x)^t$, $\bar{F} = (\bar{f}\ 1)^t$, and $z_{n+1}(x_0) = x_0$. The latter forms remove the explicit dependence on x from $\bar{f}$; we shall use the nonautonomous forms (1.2) and (1.3) throughout.

Since the solution of an IV problem invovling a system of first-order equations is, in principle, no more difficult than solving a single first-order equation, it is sufficient to study algorithms for solution of the scalar equation with initial condition

$$\frac{dy}{dx} = y'(x) = f(x,y), \quad y(x_0) = y_0. \tag{1.5}$$

The commonly used IV methods for solution of (1.5) are stepping algorithms that start with the initial condition and generate approximations y_i of the dependent variable $y(x_i)$ at discrete values of the independent variable, x_i, i=1,2,...,n, with

$$x_{i+1} = x_i + h_i, \tag{1.6}$$

where h_i is the stepsize for the ith step. Often, all h_i are equal in length, and the independent variable coordinates are

$$x_i = x_0 + ih, \quad i=1,2,\ldots,n, \tag{1.7}$$

where the overall interval of integration is (x_0, x_n), as shown in Figure 1.

In the absence of round-off error, the discrepancy between y_i and

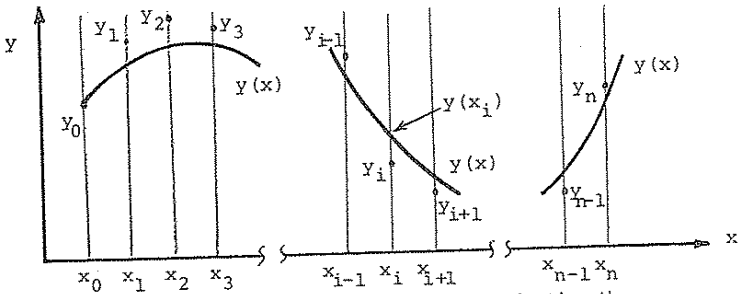

FIGURE 1 Numerical solutions y_i approximating the true solution y(x) on the interval (x_0, x_n)

$y(x_i)$, i.e., between the computed approximation and the true solution, is called the global truncation or discretization error

$$\varepsilon_i = y_i - y(x_i)\,, \tag{1.8}$$

and is determined solely by the nature of the approximations in the method. The change in gobal truncation error occurring in a single application of one of the stepping algorithms consists of two components, a propagated error resulting from approximations produced on previous steps and a local truncation error, which is the error that would be observed if all previously determined variable values were error free (i.e., if ε_i were zero for all previous steps). For sufficiently small stepsize h, the change in global truncation error on one step is normally dominated by the local truncation error whose principal component has the form Kh^{p+1} (K usually a function of derivatives of f). The local truncation error is said to be of order p+1 $[O(h^{p+1})]$ and the algorithm is called a pth order method; this implies that for a solution function of the form $y(x) = x^p$, the local truncation error will vanish.

RUNGE-KUTTA METHODS

The simplest (and oldest) algorithm for solution of the IV problem

is attributed to Euler and has the form

$$y_{i+1} = y_i + hf(x_i,y_i)\,, \tag{1.9}$$

with local truncation error $(h^2/2)f'(\xi,y(\xi))$, $x_i < \xi < x_{i+1}$, which follows directly from the Taylor's expansion of y(x) about the solution point $[x_i,y(x_i)]$. In the absence of roundoff, the gobal truncation error at x_{i+1} is

$$\varepsilon_{i+1} = \varepsilon_i[1+h(\frac{\partial f}{\partial y})_{x_i,\alpha}] - \frac{h^2}{2} f'(\xi,y(\xi))\,, \tag{1.10}$$

with α in $(y_i,y(x_i))$ and ξ in (x_i,x_{i+1}). Here, the first term corresponds to the propagation of errors in y_i and the second to the local truncation error. Additionally, the method is <u>convergent</u>, that is, the computed solution tends to the true solution in the limit as h approaches zero. The global truncation error is O(h). [It is in general true for all the stepping algorithms that the local and gobal truncation errors for a pth order method are $O(h^{p+1})$ and $O(h^p)$, respectively.]

Euler's method has the simple geometric interpretation shown in Figure 2. One simply approximates the integral in

$$y_{i+1} = y(x_i) + h\int_{x_i}^{x_{i+1}} f(x,y(x))dx\,, \tag{1.11}$$

by assuming that f(x,y(x)) is constant over the interval and has the value $f(x_i,y_i)$. For most ODE, Euler's method is simply not accurate enough for reasonable stepsize h; equivalently, it requires far too many steps [evaluations of f(x,y)] to solve the ODE over an integration interval of any substantial length.

One approach to improving the accuracy of Euler's method would be to include in the algorithm higher-order terms in the Taylor's expansion

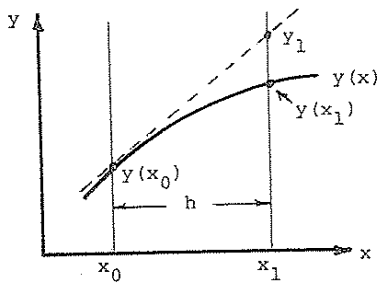

FIGURE 2 Euler's Method

of the solution function,

$$y(x_{i+1}) = y(x_i) + hf(x_i,y(x_i)) + \frac{h^2}{2} f'(x_i,y(x_i)) + \ldots. \tag{1.12}$$

Unfortunately, this approach turns out to be impractical for most ODE because the higher-order total derivatives of f(x,y) must be determined by chain rule differentiation. In order to avoid this, Runge and Kutta developed a family of methods (henceforth called RK methods) that require the calcuation of first derivatives f(x,y) only, but that have accuracies comparable to (1.12) with higher order terms retained. All of their methods are explicit in y_{i+1} (i.e., y_{i+1} can be computed directly, without iteration) and have the appearance

$$y_{i+1} = y_i + h\emptyset(x_i,y_i,h)\,. \tag{1.13}$$

$\emptyset$ is called the <u>increment</u> function and is a weighted average value of f(x,y(x)) computed at more than one (say r) points on the interval $[x_i,x_{i+1}]$; for methods of order less than five, a pth order method requires p evaluations of f on the interval. For example, the second-order explicit RK methods have the form

$$y_{i+1} = y_i + ak_1 + bk_2\,, \tag{1.14a}$$

where

$$k_1 = hf(x_i, y_i),$$
$$k_2 = hf(x_i + ph,\ y_i + qk_1). \tag{1.14b}$$

The parameters a, b, p, and q in (1.14) are established by expanding k_2 in a two-variable Taylor's series, substituting for k_1 and k_2 in (1.14a), and requiring agreement with the Talyor's expansion of y(x) in (1.12) for terms in h^0, h^1, h^2. The conditions imposed by these equivalences yield three simultaneous equations in the four parameters a, b, p, and q, one of which can be chosen arbitrarily. This leads to a family of second-order RK methods, two of which are:

$$y_{i+1} = y_i + \tfrac{1}{2}(k_1 + k_2),$$
$$k_1 = hf(x_i, y_i), \tag{1.15a}$$
$$k_2 = hf(x_i + h,\ y_i + k_1),$$

and

$$y_{i+1} = y_i + k_2,$$
$$k_1 = hf(x_i, y_i), \tag{1.15b}$$
$$k_2 = hf(x_i + \tfrac{h}{2},\ y_i + \tfrac{1}{2}k_1).$$

Higher-order methods can be developed in similar fashion. The most-used explicit RK methods are the fourth-order algorithms attributed to Runge and Gill (1), respectively:

$$y_{i+1} = y_i + \tfrac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$
$$k_1 = hf(x_i, y_i),$$
$$k_2 = hf(x_i + \tfrac{1}{2}h,\ y_i + \tfrac{1}{2}k_1), \tag{1.16a}$$
$$k_3 = hf(x_i + \tfrac{1}{2}h,\ y_i + \tfrac{1}{2}k_2),$$
$$k_4 = hf(x_i + h,\ y_i + k_3),$$

and

$$y_{i+1} = y_i + \tfrac{1}{6}(k_1 + 2c_1k_2 + 2c_2k_3 + k_4),$$
$$k_1 = hf(x_i, y_i),$$
$$k_2 = hf(x_i + \tfrac{1}{2}h,\ y_i + \tfrac{1}{2}k_1),$$
$$k_3 = hf(x_i + \tfrac{1}{2}h,\ y_i + (c_2 - \tfrac{3}{2})k_1 + c_1k_2), \tag{1.16b}$$
$$k_4 = hf(x_i + h,\ y_i + (c_1 - 1)k_2 + c_2k_3),$$
$$c_1 = (1 - 1/\sqrt{2}),$$
$$c_2 = (1 + 1/\sqrt{2}).$$

Higher order classical explicit RK methods are also available; in most cases, pth-order methods for p greater than four require more than p evaluations of f(x,y) on $[x_i, x_{i+1}]$. Methods of order 5 and 6 attributed to Butcher (2), Luther (3,4,5) and Felhberg (6) are the most used of these higher-order formulas.

In 1963, Rosenbrock (7) proposed a more general family of RK algorithms of the form:

$$y_{i+1} = y_i + \sum_{j=1}^{r} w_j k_j, \tag{1.17a}$$
$$k_j = hf(x_i + c_j h,\ y_i + \sum_{\ell=1}^{r} a_{j\ell} k_\ell), \tag{1.17b}$$
$$c_1 = 0,$$
$$c_j = \sum_{\ell=1}^{r} a_{j\ell}, \qquad \sum_{j=1}^{r} w_j = 1.$$

The various parameters of the methods can be laid out in a compact form called a Butcher (8) block as is shown in (1.18).

If matrix A is strictly lower triangular (no elements on or above the diagonal), then (1.17) will involve r evaluations of the k_j, in which the k_j appear explicitly, and the k_j can be computed in order $k_1, k_2, \ldots, k_r$. On the other hand, if A is merely lower triangular

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1r} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & & & & \\ c_r & a_{r1} & a_{r} & \cdots & a_{rr} \\ \hline & w & w_2 & \cdots & w_r \end{array} \quad \text{or} \quad \begin{array}{c|c} \bar{c} & A \\ \hline & \bar{w}^t \end{array} \qquad (1.18)$$

(some nonzero elements on the diagonal), (1.17b) leads to r simultaneous equations in the k_j, in which the k_j appear implicitly. Because the equation in k_1 involves only k_1, that in k_2 involves only k_1 and k_2, etc., the k_j can still be computed in the order k_1, k_2, ... k_r (provided the equations can be solved), and the corresponding method is called a semi-implicit RK method. Assuming that h is small, the k_j can be expanded in Taylor series, and approximated by the linear terms only, in which case the implicit equations are linear and can be solved easily for the k_j. If A is full, then (1.17b) leads to a system of r implicit equations in the k_j that must be solved simultaneously (i.e., cannot be solved sequentially, as in the semi-implicit cases); the corresponding algorithms are called fully-implicit RK methods.

For the system of simultaneous ODE of (1.3), the vector form of (1.17) for the semi-implicit methods is:

$$\bar{y}_{i+1} = \bar{y}_i + \sum_{j=1}^{r} w_j \bar{k}_j , \qquad (1.19a)$$

$$\bar{k}_j = h\bar{f}(x_i + c_j h, \bar{y}_i + \sum_{\ell=1}^{j} a_{j\ell}\bar{k}_\ell) \qquad (1.19b)$$

If the $\bar{k}_j$ are linearized using Taylor expansions, (1.19b) assumes the

form

$$\bar{k}_j = h(I - a_{jj}hJ_j)^{-1}\bar{f}(x_i + c_j h, \bar{y}_i + \sum_{\ell=1}^{j-1} a_{j\ell}\bar{k}_\ell) , \qquad (1.20)$$

where $J_j = J(x_i + c_j h, \bar{\xi}_j)$, $\bar{\xi}_j$ in $(\bar{y}_i, \bar{y}_{i+1})$, is the Jacobian of the ODE system

$$J = \frac{\partial f_s}{\partial y_q} . \qquad (1.21)$$

Equations (1.20) represent r systems of n simultaneous linear equations that can be solved sequentially for $\bar{k}_1$, $\bar{k}_2$, ..., $\bar{k}_r$. Several researchers, in particular Rosenbrock (7), Calahan (9), Caillaud (10), Michelsen (11), Cash (12), Ballard and Brosilow (13), and Chan (14) have developed semi-implicit RK methods of orders p = 2, 3, and 4 requiring $r \leq p$ function evaluations and usually only one Jacobian evaluation per step. These methods possess very good stability characteristics as described in later subsections and are suitable for solving systems of stiff ODE (as described later in the paper).

The fully implicit RK methods for n simultaneous ODE (for which the upper limit on the summation in (1.19b) is r rather than j) lead to systems of nr simultaneous nonlinear equations. Because of their computational complexity, they are seldom used.

The fourth-order explicit methods of (1.16) have been found suitably accurate for most non-stiff first-order systems of ODE, and virtually every computing library includes a general purpose subroutine for thier implementation. The RK methods were the most popular IV methods during the 1950's and early 1960's because of the following advantages over the linear multistep methods of the next subsection:

1. They are self-starting; only the differential equations and their initial conditions are required for the first application of a

Runge-Kutta method.

2. Since every new step can be viewed as the beginning of a new IV problem, the step-size can be changed easily for the next application of the method.

3. No iteration (even for the semi-implicit RK methods) is required.

4. The methods are easy to program, and require little storage of historical information (past values of the y_i and f_i).

5. They produce results of good accuracy, often better than the linear-multistep methods for comparable stepsizes.

The RK methods have two principal disadvantages:

1. It is difficult to estimate the local truncation error and hence to establish criteria for automatically adjusting the stepsize.

2. The linear multistep methods require significantly fewer function (derivative) evaluations per step than do the RK methods. [Typically, the commonly-used fourth-order RK methods require twice as many function evaluations as the most-used of the fourth-order linear multistep algorithms.]

The local truncation error term for a pth order RK method has the form

$$\epsilon_t = Kh^{p+1} + O(h^{p+2}), \tag{1.22}$$

where K is a vary complicated (and essentially incalculable) function of f(x,y) and of its higher-order derivatives (1.18). If we assume that K changes little from one step to the next, then Richardson's extrapolation technique (19) can be used to estimate ϵ_t. Here, the RK method is used to integrate between two points, say x_i and x_{i+2}, using first two steps of length h_1 and then one step of length $h_2 = 2h_1$. Let

$y_{i+2,1}$ and $y_{i+2,2}$ be the calculated solutions using stepsizes h_1 and h_2, respectively. Then an estimate of the truncation error for $y_{i+2,1}$ is given by

$$\epsilon_t = \frac{(y_{i+2,1} - y_{i+2,2})}{2 - 2^{p+1}} . \tag{1.23}$$

If the error is monitored on every other step (h_1), the computational burden is very severe, adding p-1 functional evaluations per test (k_1 is identical in the two cases).

Alternatively, several researchers have developed "imbedded" RK algorithms of order p and p+1, respectively, that use the same derivatives along the way, i.e., the higher-order method uses the k_j from the lower-order method, plus one or more additional derivatives. The truncation error for the lower-order method is then estimated as the difference between the computed results for the two different algorithms. The best-known of the explicit RK imbedded algorithms are probably those attributed to Sarafyan (17) and to Fehlberg (18). More recently, Chan, et al. (14) have developed several families of imbedded semi-implicit RK formulas. Many of the most useful of Chan's imbedded algorithms (designed for solving stiff systems of equations) are of order p and p+2 rather than p and p+1; it is not clear how one relates the difference in solution values computed by two such formulas to estimate the truncation error being committed. Cash (12) has developed interesting second and third order semi-implicit methods in which the imbedded forms are of the same order; the two forms are used in a one-step, two-step fashion allowing the equivalent of a Richardson's extrapolation estimate of the truncation error.

Because of the additional computational work required for error

estimation, and especially because of the large number of derivative evaluations required per step for methods producing acceptable accuracy, the explicit RK methods are now less popular than the linear multistep methods. Certain of the semi-implicit RK methods (those suitable for integration of stiff equation systems) would appear to have appeal, but thus far have not been used very widely.

LINEAR MULTISTEP METHODS

The second major class of IV methods for ODE are the linear multistep (LMS) methods of the form

$$y_{i+1} = \sum_{j=0}^{k} \alpha_j y_{i-j} + h \sum_{j=-1}^{k} \beta_j f_{i-j} , \qquad (1.24)$$

where $f_{i-j} = f(x_{i-j}, y_{i-j})$. The methods are not self-starting (except for the sepcial case where k=0), and it is usually necessary to use k applications of a single-step method to establish the required y and f values before (1.24) can be used for the first time. When $\beta_{-1} = 0$, (1.24) is explicit in y_{i+1} and the algorithm is called an open or predictor method. When $\beta_{-1} \neq 0$, (1.24) is implicit in y_{i+1}, and the algorithm is called a closed or corrector method; the equation must then be solved by an iterative technique.

Some of the most common algorithms of closed and open type (particularly those for which $\alpha_q \neq 0$, $\alpha_j = 0$, $j \neq q$) can be developed easily by fitting the f_{i-j}, $j = -1, \ldots, k$, with an interpolating polynomial and evaluating y_{i+1} from

$$y_{i+1} = y_{i-q} + \int_{x_{i-q}}^{x_{i+1}} \theta_r(x)dx . \qquad (1.25)$$

$\theta_r(x)$ is an rth degree interpolating polynomial. Given certain combinations of q and r, (1.25) leads to algorithms that are essentially

Newton-Cotes open and closed quadrature formulas. For example, Milne's fourth-order predictor (P) and corrector (C) are

$$\text{P:}\quad y_{i+1} = y_{i-3} + \frac{4h}{3}(2f_i - f_{i-1} + 2f_{i-2}) , \qquad \varepsilon_t = \frac{14}{45}h^5 f^{(4)}(\xi) , \qquad (1.26a)$$

$$\text{C:}\quad y_{i+1} = y_{i-1} + \frac{h}{3}(f_{i+1} + 4f_i + f_{i-1}) , \qquad \varepsilon_t = \frac{-1}{90}h^5 f^{(4)}(\xi) , \qquad (1.26b)$$

and their local truncation error terms ε_t follow directly from the three-point open and closed Newton-Cotes formulas, respectively (Equation (1.26b) is just Simpson's Rule). Figure 3 shows the situation for (1.26b); the shaded area is the integral of (1.25) evaluated using Simpson's Rule.

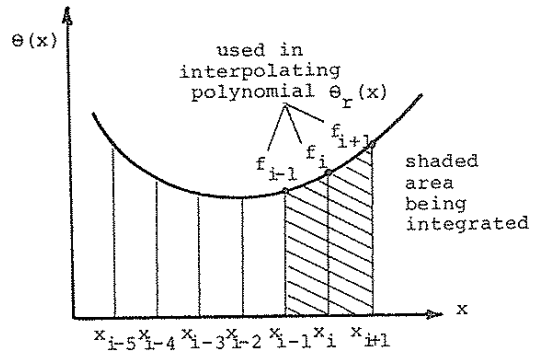

FIGURE 3 Integral of (1.25) for Milne's Corrector

The simplest and most general way of generating any LMS algorithm is to use the method of undetermined coefficients. In this approach, one chooses the number of nonzero coefficients α_j and β_j (say p+1), and requires that the formula be exact for y(x) a polynomial of degree p or less. For example, Milne's corrector could be found by requiring that

the five-constant LMS method of the form

$$y_{i+1} = \alpha_1 y_{i-1} + h(\beta_{-1} y_{i+1} + \beta_0 f_i + \beta_1 f_{i-1} + \beta_2 f_{i-2}), \quad (1.27)$$

produce exactly the correct solutions for function of the form $y(x) = 1$, x, x^2, x^3, and x^4; in this case, these conditions lead to the coefficients of (1.26b) with $\beta_2 = 0$. The local truncation error term can be found by assuming that it has the form

$$\varepsilon_t = Ch^{p+1} f^{(p)}, \quad (1.28)$$

and establishing the value of C (-1/90 in this case) by substituting $y(x) = x^5$ into (1.27); for Milne's 4th order corrector, p=4.

Other formulas of the LMS type can be generated by choosing a pth order method with more than p+1 (say m) coefficients α_j and β_j being nonzero. In general, m-p-1 of these nonzero coefficients can be chosen arbitrarily, and the remaining ones can be established using the method of undetermined coefficients. The corresponding truncation error term can usually be assumed to be of the form of (1.28), although for certain choices of the arbitrary coefficients, the method of undetermined coefficients may lead to incorrect results. (This latter problem is related to the influence function (see Hamming (14) for details) and is not a problem for the commonly-used LMS methods.) Hamming (15) used this approach in developing his fourth-order corrector

$$y_{i+1} = \frac{9}{8} y_i - \frac{1}{8} y_{i-2} + \frac{3}{8} h(f_{i+1} + 2f_i - f_{i-1}), \quad (1.29)$$

$$\varepsilon_t = \frac{1}{40} h^5 f^{(4)}(\xi),$$

starting with a seven parameter model including α_0, α_1, α_2, β_{-1}, β_0, β_1, β_2. Hamming studied the stability characteristics (see the next subsection) as a function of the two free parameters and selected the coefficient set of (1.29) to produce a method having a fairly small

value of C in (1.28), good error propagation properties, and zero values for two of the seven coefficients, α_1 and β_2.

The most popular of the open multistep methods are probably those from the Adams-Bashforth family of predictors of the form

$$y_{i+1} = y_i + h \sum_{j=0}^{k} \beta_j f_{i-j}. \quad (1.30)$$

The coefficients β_j, $j=0,1,\ldots,k$ (scaled by a factor s) and C (in (1.28)) for methods of order 1 through 6 are shown in Table 1. For example, β_2 for the fourth-order method is 37/24 and the local truncation error term is $(251/720)h^5 f^{(4)}(\xi)$.

TABLE Coefficients for the Adams-Bashforth Predictors

Order p	Scale Factor s	β_0	β_1	β_2	β_3	β_4	β_5	C
1	1	1						1/2
2	2	3	-1					5/12
3	12	23	-16	5				9/24
4	24	55	-59	37	-9			251/720
5	720	1901	-2774	2616	-1274	251		475/1440
6	1440	4277	-7923	9982	-7298	28771	-475	19087/60480

The first-order method is just the single-step Euler's method of (1.9).

The corresponding closed Adams-Moulton correctors of the form

$$y_{i+1} = y_i + h \sum_{j=-1}^{k} \beta_j f_{i-j}, \quad (1.31)$$

are among the most popular corrector equations; Table 2 lists the coefficients for orders 2 through 6. The second-order Adams-Moulton corrector is more commonly called the trapezoidal rule. A large number of other multistep predictors and correctors can be found in the excellent text by Lapidus and Seinfeld (17).

TABLE 2 Coefficients for the Adams-Moulton Correctors

Order	Scale Factors	β_{-1}	β_0	β_1	β_2	β_3	β_4	C
2	2	1	1					-1/12
3	12	5	8	-1				-1/24
4	24	9	19	-5	1			-19/720
5	720	251	646	-264	106	-19		-27/1440
6	1440	475	1427	-798	482	-173	27	-863/60480

The corrector equation of (1.31) is implicit in y_{i+1} and hence must be solved by an iterative approach. Straightforward successive substitution is normally used, in which case (1.31) has the form

$$y_{i+1,j+1} = h\beta_{-1}f(x_{i+1},y_{i+1,j}) + K \tag{1.32}$$

where K includes all of the known terms on the right-hand-side of (1.31) and j is the iteration counter. The criterion for convergence of (1.32) is

$$h < 1/\left|\beta_{-1}\frac{\partial f}{\partial y}\right|_{x_{i+1},y_{i+1}} \tag{1.33}$$

For rapid convergence (essential if the number of derivative evaluations is to be kept small), h should probably be no larger than one tenth of the bound given by (1.33). For systems of equations of the form of (1.3), convergence of the vector analog of (1.31) by Jacobi or Gauss Seidel iteration (see p. 83) requires that h satisfy the criterion

$$h < 1/|\beta_{-1}\lambda_{max}|, \tag{1.34}$$

where λ_{max} is the largest (in magnitude) eigenvalue of the Jacobian of the equation system. Should λ_{max} be quite large, it may be necessary to resort to Newton-Raphson or quasi-Newton methods to solve the implicit nonlinear equations if reasonable stepsizes h are to be used (frequently the case for stiff equation systems).

Comparison of the coefficient C for the Adams predictor and corrector of the same order shows that, in general, implicit formulas are considerably more accurate than explicit formulas. To take advantage of the superior accuracy of a closed formula and at the same time to reduce the number of iterations required for convergence, an open formula of identical order is normally used to produce the first estimate, $y_{i+1,0}$, for the closed formula. Such algorithms, employing an explicit predictor equation for the first guess, and an implicit corrector equation for generating the final numerical solution, y_{i+1}, are called predictor-corrector methods.

As an example of a predictor-corrector method, consider Hamming's method, which uses the Milne predictor (1.26a) and the Hamming corrector (1.29). Normally, one would begin with three applications of a single-step method of comparable accuracy (say a fourth-order RK method) to generate the essential parameters $(y_1,f_1,y_2,f_2,y_3,f_3)$ before the first application of the predictor equation in the predictor-corrector (PC) algorithm. Then the procedure is as follows, beginning with i=3:

1. Calculate $y_{i+1,0}$ using the predictor (1.26a)
2. Compute $f(x_{i+1},y_{i+1,0})$ and then solve the corrector (1.29a) iteratively as in (1.32) until an absolute or relative convergence test such as

$$\left|\frac{y_{i+1,j+1} - y_{i+1,j}}{y_{i+1,j+1}}\right| < \eta, \tag{1.35}$$

is passed (here, η is a small positive number supplied by the user). Let $y_{i+1,k}$ be the converged solution.

3. Increment i and repeat from step 1.

Normally, estimates of the local truncation errors are computed.

For the LMS methods, this turns out to be very simple, and computationally inexpensive. Using the local error terms (1.26a) and (1.29) for predictor and corrector, respectively, a Richardson's extrapolation leads to the following estimates, ϵ_{t_c} (corrector) and ϵ_{t_p} (predictor):

$$\epsilon_{t_c} \doteq -\frac{9}{121}(28y_{i+1,k} - y_{i+1,0}) , \tag{1.36a}$$

$$\epsilon_{t_p} \doteq \frac{112}{121}(28y_{i+1,k} - y_{i+1,0}) . \tag{1.36b}$$

These truncation error estimates can be used for adjusting the stepsize, given user specified upper and lower bounds for the allowable error. In addition, assuming that the truncation error for the predictor does not change appreciably from one step to the next, the predicted value $y_{i+1,0}$, can be modified to produce a better first guess for the corrector

$$y^*_{i+1,0} = y_{i+1,0} + \epsilon_{t_{p,i}} , \tag{1.37a}$$

where $\epsilon_{t_{p,i}}$ is the predictor error estimate from the preceding step. This has no effect on the corrector, except to reduce (usually) the number of iterations required for convergence. In fact, the stepsize h is normally chosen to permit convergence of the corrector in just <u>one</u> iteration. In this predictor-modified predictor-corrector algorithm, the derivative f is evaluated just <u>twice</u> per step; this is only half the number required by the explicit fourth-order RK methods. In addition, the value $y_{i+1,k}$ produced by the corrector (presumably iterated to convergence or near convergence) can itself be modified as

$$y_{i+1} = y_{i+1,k} + \epsilon_{t_c} , \tag{1.37b}$$

leading to a four step algorithm: (1) predict, (2) modify the predicted value, (3) correct, and (4) modify the corrected value. Some

authors feel that the latter modification should not be made as it may affect the stability (see the next subsection) of the corrector, but some available library programs do implement all four steps.

Disadvantages of the LMS method are:

1. They are not self-starting; the methods must be started using a single-step RK method of comparable order, or a low order LMS method with small stepsize.

2. The step-size is not as easily changed as in the RK methods. Variable stepsize versions of all the LMS methods can be generated, but they are more complex computationally than the fixed-stepsize versions of the equations.

3. Because of the implicit nature of the corrector equations, some form of iteration is required for their solution.

4. The methods are more complicated to program than are the RK methods (though still not particularly difficult) and usually require storage of historical information (past values of the y_i and f_i).

On the positive side are the very desirable features of simple local truncation error estimation and substantially reduced number of function (derivative) evaluations required per step (compared with RK methods). In the last decade, these positive factors, and the reduced computational costs resulting from them, have made the LMS methods the most popular algorithms for both stiff and nonstiff problems.

<u>STABILITY</u>

Ideally, one would like to choose an IV algorithm and a stepsize adjustment strategy that will produce the required accuracy with minimum computing cost. If the derivatives are at all complicated, then the computing cost will be directly proportional to the number of function evaluations per step and the number of steps assuming that

the "bookkeeping" operations cost relatively little. The stepsize adjustment is usually controlled by estimates of the local truncation error as described in the preceding two subsections, assuming that the local error dominates the incremental error on each step. Is such an assumption justified?

To gain some insight into the nature of this problem, consider the global truncation error ε_{i+1} for Euler's method (1.10)

$$\varepsilon_{i+1} = \varepsilon_i\left[1+h\left(\frac{\partial f}{\partial y}\right)_{x_i,\alpha}\right] - \frac{h^2}{2} f'(\xi,y(\xi))$$

with α in $(y_i,y(x_i))$ and ξ in (x_i,x_{i+1}). Clearly, if the propagated error from the first term dominates the local truncation error of the second, any assumptions about the size of the error based on the approaches of the preceding sections would be invalid; in fact, one can see that the propagated error might be so large as to render the numerical solution meaningless.

To illustrate possible problems, consider the solution of the scalar linear ODE

$$\frac{dy}{dx} = \lambda y \quad \text{with } y(0) = 1, \tag{1.38}$$

for which the true solution is

$$y(x) = e^{\lambda x}. \tag{1.39}$$

Substituting (1.39) into (1.10) yields

$$\varepsilon_{i+1} = \varepsilon_i(1+h\lambda) - \frac{(h\lambda)^2}{2} e^{\lambda\xi}, \tag{1.40}$$

with ξ in (x_i,x_{i+1}). If λ is positive (assume h is positive), then the error ε_i will be amplified by the "propagation factor" $(1+h\lambda)$. This may not be disasterous, since $y(x)$ itself is growing without bound. However, note the difficulty when λ is negative. If $(1+h\lambda)$ is greater than one in magnitude, then the error will grow exponentially, alternating in sign from step to step, while the true solution is decaying exponentially. We would hope that any generated errors would decay along with the solution, which will happen only if

$$|1+h\lambda| < 1. \tag{1.41}$$

The situation may be somewhat clearer if (1.38) is incorporated directly into the Euler's method algorithm of (1.9)

$$y_{i+1} = y_i + hf(x,y) = y_i + h\lambda y_i, \tag{1.42}$$

or

$$y_{i+1} = (1+h\lambda)y_i = y_0(1+h\lambda)^i = (1+h\lambda)^i. \tag{1.43}$$

Note that the solution will be valid only to the extent that $(1+h\lambda)^i$ is a reasonable approximation of $e^{ih\lambda} = e^{\lambda x}$. If λ is negative, and the stepsize is chosen to assure that (1.41) holds, then (1.43) will decay in magnitude; if we want monotonically decaying behavior, the even more restrictive condition

$$h < \left|\frac{1}{\lambda}\right| \tag{1.44}$$

must hold. Thus for the simple model equation, Euler's method has a restricted range of values for $h\lambda$ when λ is negative (when the true solution is decaying). If, for a given negative λ too large a stepsize h is used, errors will propagate without bound and swamp the true solution to the ODE. The method will be unstable.

Although numerous types and definitions of stability appear in the literature, we are here concerned with two types, relative and absolute. Relative stability is of importance when the true solution of the ODE is growing (for example, the solution of (1.38) with positive λ); if the method is to be stable, then any errors introduced (in the initial condition or subsequently as a result of truncation or roundoff errors)

must grow no more rapidly in the relative sense than does the true solution. Absolute stability is critical when the true solution is decaying (for example, the solution of (1.38) with negative λ). Any errors introduced must decay along with the true solution; otherwise, the propagated error may overwhelm the true solution.

An IV method cannot be considered an acceptable candidate for solution of an ODE unless it is stable and possesses properties called consistency and convergence (see (20)). A method is convergent if

$$\lim_{\substack{h \to 0 \\ n \to \infty}} y_n = y(x_n), \quad hn = x_n - x_0 . \tag{1.45}$$

Thus, a convergent method is one which in the limit as h approaches zero, yields a numerical solution that approaches the true solution. Consistency, a necessary condition for convergence, is equivalent to requiring that the method be exact for y(x) a linear polynomial. For the LMS methods, the conditions

$$\begin{aligned} \sum_{j=0}^{k} \alpha_j &= 1 , \\ \sum_{j=0}^{k} j\alpha_j - \sum_{j=-1}^{k} \beta_j &= -1 , \end{aligned} \tag{1.46}$$

must hold (with the α_j and β_j from (1.24)). Similarly, for RK methods, the conditions of (1.17) must be satisfied.

One of the first stability analyses of an LMS method was performed on the Milne corrector (1.26b), assuming that the corrector is iterated to convergence (if the corrector in a PC method is not iterated to convergence, then the stability analysis must include both predictor and corrector as shown in (17)). Substituting (1.38) into (1.26b) leads to a system of second-order linear homogeneous difference equations with

constant coefficients

$$(1-\tfrac{\lambda h}{3})y_{i+1} - (\tfrac{4\lambda h}{3})y_i - (1+\tfrac{\lambda h}{3})y_{i-1} = 0 . \tag{1.47}$$

Assuming solutions of the form $y_i = \gamma^i$ (21), and substituting into (1.47) leads to the characteristic equation

$$(1-\tfrac{h\lambda}{3})\gamma^2 - (\tfrac{4h\lambda}{3})\gamma - (1+\tfrac{h\lambda}{3}) = 0 , \tag{1.48}$$

which has two distinct roots (using the quadratic formula). When expanded in power series (22), the two roots have the form

$$\begin{aligned} \gamma_1 &= e^{\lambda h} + \frac{(\lambda h)^5}{180} + \cdots \\ \gamma_2 &= -1 + \frac{(\lambda h)}{3} - \frac{(\lambda h)^2}{18} - \frac{(\lambda h)^3}{54} + \frac{5(\lambda h)^4}{648} + \frac{5(\lambda h)^5}{1944} + \cdots \end{aligned} \tag{1.49}$$

In a manner analogous to that for ODE's, the solution of (1.47) is

$$y_i = c_1\gamma_1^i + c_2\gamma_2^i . \tag{1.50}$$

Incorporating (1.49) into (1.50) yields

$$y_i = c_1 e^{ih\lambda} + c_2\gamma_2^i = c_1 e^{\lambda x} + c_2\gamma_2^i . \tag{1.51}$$

Thus the first root, γ_1, called the principal root, generates an approximation to the true solution. Unfortunately, the quantity $c_2\gamma_2^i$, unrelated to the true solution of the ODE, also contributes to the numerical solution. This second root is called parasitic or spurious and appears because the first-order ODE is being approximated by a second-order difference equation. Whether the method is stable or not depends on the relative contributions of the two terms, and as we shall see, this depends in turn on the sign of λ.

Figure 4 shows the values of the two roots as a function of λh in (-1,1). This range of λh is the one of interest since the criterion for convergence of the corrector iteration (1.33) requires that $|h\lambda|<3$, and for rapid convergence, the range should be considerably smaller.

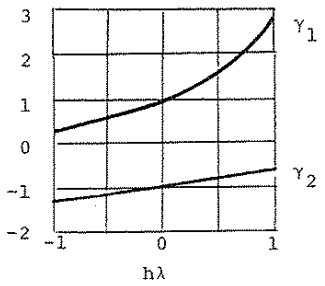

FIGURE 4 Roots of the characteristic equation of the Milne corrector (1.26b) for the model problem (1.38)

Note that for positive λ, the principal root produces an exponentially growing component corresponding to the true solution, while the parasitic solution decays (with alternating sign from step to step). The method will be stable. On the other hand, for negative λ, the principal root produces an exponentially decaying component corresponding to the true solution, while the parasitic solution grows without bound (alternating in sign from step to step), eventually masking the true solution. In this latter case, the method is unstable; in fact, decreasing the stepsize aggravates the stability problem.

A similar stability analysis for the LMS methods of (1.24) with the model problem (1.38) leads to the kth-order difference equations

$$\sum_{j=-1}^{k} (\alpha_j + h\lambda\beta_j) y_{i-j} = 0 , \qquad (1.52)$$

with $\alpha_{-1} = -1$. The characteristic equations for (1.52) will posses k roots, one of which corresponds to the true solution (the principal root, γ_1); the remaining k-1 roots will be parasitic. A necessary condition for convergence of a method is that no root of its characteristic equation with h=0 lie outside the unit circle, and that roots of magnitude 1 be simple (18). The principal root for the h=0 case always has a magnitude of one. Assuming that all roots of the characteristic equation are distinct, the numerical solution will have the form

$$y_i = \sum_{j=1}^{k} c_j \gamma_j^i . \qquad (1.53)$$

In the limit, as h approaches zero, c_1 in (1.53) approaches $y(x_0)$, while $c_2, c_3, \ldots, c_k$ approach zero.

For finite but small h, the coefficients $c_2, \ldots, c_k$ should be small, but if the parasitic solutions are to remain small relative to the principal solution $c_1\gamma_1^i$, then it is essential that

$$|\gamma_1| \geq |\gamma_j| \quad , \quad j=2,\ldots,k. \qquad (1.54)$$

Since $\gamma_1 \to 1$ as $h \to 0$, all roots of the characteristic equation when h=0 must lie on or within the unit circle (here, we assume that λ may be complex).

For finite h, the roots may be outside the unit circle. Two cases are of interest. If the real part of λ is positive, then $|\gamma_1| > 1$, and if the principal solution is to dominate the parasitic solutions, (1.54) must hold. The method is said to be relatively stable. If the real part of λ is negative, then

$$|\gamma_j| < 1, \quad j=1,2,\ldots,k \qquad (1.55)$$

must hold; this condition is called absolute stability. Criterion (1.55) simply insures that all parasitic components decay along with the principal component when the true solution is decaying.

All of the classical LMS methods have been analyzed for their

regions of absolute stability for the model equation; the troublesome instabilities for most methods occur when the true solution decays. Since the true solution decays when the real part of λ is negative (imaginary parts of λ produce oscillations about the decaying solution, but do not alter its essential character), stability plots for the various methods focus on the left half of the complex $h\lambda$ plane (see Figure 6, for example) for which $Re(h\lambda) < 0$.

Even when there is no parasitic solution present, the absolute stability criterion of (1.55) for the one (and principal) root may substantially restrict the range of $h\lambda$ that may be used. For example, the characteristic equation for the linear difference equation for Euler's method (1.42) and the model problem (1.38) is

$$\gamma - (1 + h\lambda) = 0, \tag{1.56}$$

and the single root is

$$\gamma = (1 + h\lambda).$$

The absolute stability criterion of (1.55) leads directly to (1.41). The root is plotted in Figure 5. The region of absolute stability for complex $h\lambda$ is shown in Figure 6.

It is instructive to examine the second-order Adams corrector (the trapezoidal rule) for stability. The first-order difference equation for the model problem has the characteristic equation

$$\gamma(1 - \frac{h\lambda}{2}) - (1 + \frac{h\lambda}{2}) = 0, \tag{1.57}$$

with the single root

$$\gamma = (1 + \frac{h\lambda}{2})/(1 - \frac{h\lambda}{2}),$$

and is absolutely stable over the entire left-half of the complex $h\lambda$ plane. A method possessing this property is called A stable. It is important to note that in the limit as $h\lambda \to -\infty$, $\gamma \to -1$. While the

trapezoidal rule remains stable for $Re(h\lambda)$ large and negative, the stability limit is being approached. The numerical solution of the model problem will not decay to zero for large stepsizes when λ is large and negative and will oscillate to boot.

Yet another interesting low-order method is the first-order backward Euler's method,

$$y_{i+1} = y_i + hf(x_{i+1}, y_{i+1}). \tag{1.58}$$

This is an implicit algorithm (hence must be solved iteratively) and can be considered to be the first-order Adams-Moulton corrector. The characteristic equation for (1.58) has a single root,

$$\gamma = 1/(1 - h\lambda). \tag{1.59}$$

This method is A stable. In addition, $\gamma \to 0$ as $Re(h\lambda) \to -\infty$, so that the numerical solution of the model problem when $Re(\lambda)$ is large and negative decays to zero (even for large h) as does the true solution. A method with this property is said to be strongly A stable (some authors call the property L stability).

Dahlquist (23) has proved several theorems about the stability of the LMS methods including the following (here, p is the order of the method and k is the order of the difference equation):

1. A k-step LMS method can exhibit regions of absolute stability in the sense of (1.55) only if the order p is less than or equal to k+2 for k even and k+1 for k odd. Thus there is a competition between stability and accuracy.

2. No explicit method can be A stable.

3. The order p of an A stable LMS method cannot exceed 2, and the most accurate (smallest local truncation error term) of all such methods is the trapezoidal rule.

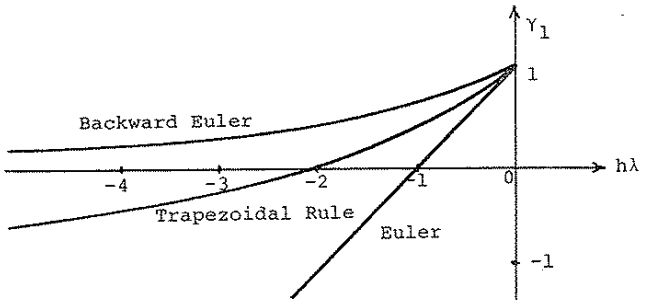

FIGURE 5 Principal Root of the Characteristic Equation for the Model Problem

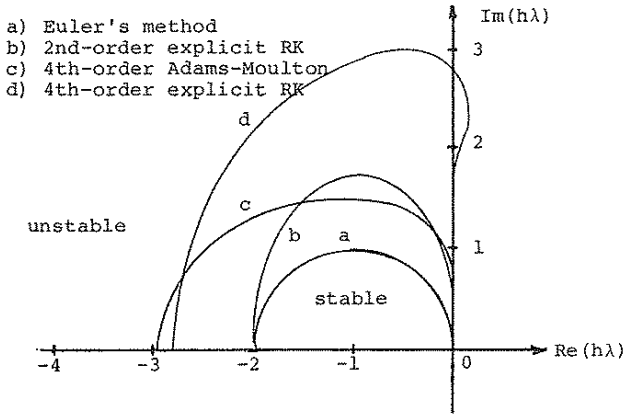

FIGURE 6 Regions of Absolute Stability for the Model Problem for Complex hλ (symmetric about the real axis)

Widland (24) subsequently proved that for λ real (actually λ can have a restricted imaginary component) A stable methods exist for which k=p=3 and 4.

The results of the work of Dahlquist are rather disappointing, since it is not possible to generate high-order LMS methods with absolute stability over the entire left half of the complex hλ plane. As we shall see in the next subsection, A-stable or nearly A stable methods are essential for the solution of systems of stiff equations.

The Runge-Kutta methods can be similarly analyzed for the model problem. Since these methods are one-step methods, the system of linear difference equations generated by substitution of (1.38) is always of order one. For example, for the algorithm of (1.15a), the first-order difference equation is

$$y_{i+1} = (1+h\lambda+h^2\lambda^2/2)y_i\,, \tag{1.60}$$

and the single root of the characteristic equation is

$$\gamma = 1+h\lambda+\frac{h^2\lambda^2}{2}\,. \tag{1.61}$$

(One can show that all the second-order explicit RK methods have identical characteristic equations and roots.) Since the RK methods have only one root, there are no parasitic solutions to contend with, and relative stability as defined by (1.54) is meaningless. All explicit RK methods do, however, have absolute stability (1.55) limits. The $\mathrm{Re}(h\lambda) \leq 0$ limits for absolute stability for all of the explicit RK methods is about -4 to 0 (see Figure 6).

Unlike the LMS methods, implicit RK methods of arbitrarily high order having A stability or strong A stability can be developed, and several papers on this topic have been published in the past ten years or so. Most of these are semi-implicit of the form of (1.19) and

follow from the early work of Rosenbrock (7), whose algorithm is third order and requires two derivative evaluations per step and two Jacobian evaluations and inversions per step (usually by LU decomposition). The method is A stable with the root approaching -0.8 for large negative Re(hλ). Calahan (9) developed a third-order method with two derivative evaluations and only one Jacobian evaluation that is A stable with the root approaching -0.732. Caillaud and Padmanabhan (10) developed a third-order strongly A stable method requiring two derivative evaluations, one Jacobian evaluation, and a matrix multiplication. Michelsen (11) modified this method by eliminating a matrix multiplication. His algorithm has the form:

$$\begin{aligned} \bar{y}_{i+1} &= \bar{y}_i + w_1\bar{k}_1 + w_2\bar{k}_2 + \bar{k}_3, \\ \bar{k}_1 &= h(I - haJ)^{-1}\bar{f}(x_i, \bar{y}_i), \\ \bar{k}_2 &= h(I - haJ)^{-1}\bar{f}(x_i + 0.75h, \bar{y}_i + 0.75\bar{k}_1), \\ \bar{k}_3 &= (I - haJ)^{-1}(b_1\bar{k}_1 + b_2\bar{k}_2). \end{aligned} \tag{1.62}$$

Chan et al. (14) have developed several semi-implicit methods involving imbedded forms varying in order from one to four. For example, they found a strongly A stable first-order imbedded form for Michelsen's method of (1.62),

$$\bar{y}_{i+1} = \bar{y}_i + w_1'\bar{k}_1 + w_2'\bar{k}_2, \tag{1.63}$$

where $\bar{k}_1$ and $\bar{k}_2$ are identical to those in (1.62).

More recently, Bui (25) has developed fourth-order RK semi-implicit methods requiring just one Jacobian inversion per step that possess strong A stability.

Ehle (26) has shown that all fully-implicit RK methods of order 2p requiring p derivative evaluations per step are A stable. Despite their

obvious attractiveness from the standpoint of stability and accuracy, these methods have not been viewed very favorably. Their principal drawback is the necessity of solving a large system of nonlinear equations at each timestep for the $\bar{k}_j$ of (1.17b). Butcher (27) has pointed out that for systems yielding a narrow band structure for the Jacobian, sparse matrix schemes may make the implicit forms viable. For the moment, however, prospects for extensive use of fully implicit RK methods seem dim. Although the semi-implicit methods do not appear to have been used as the basis for widely available software packages, they do seem to have considerable promise for solving stiff problems, where A stability is very important.

To conclude this subsection, we note that all the preceding analyses have been for a rather simple linear ODE. Since most real-world problems will be neither linear nor simple, one might question the utility of such analyses. Fortunately, as shown by Hildebrand (28), for most nonlinear ODE the nature of the error propagation process is similar to that for the linear approximation of the ODE, at least over a short range near x_i. Hence, one can expect a method to behave qualitatively as it does for the model equation with λ replaced by $\partial f/\partial y$.

STIFF EQUATIONS

Some mathematically well behaved (stable) systems of ODE cannot be solved effectively with traditional algorithms (explicit RK or Adams predictor-corrector LMS methods) because the rather limited regions of absolute stability force the use of extremely small stepsizes, or, correspondingly, an impossibly large number of steps for integration over a reasonable interval in the independent variable. Usually such systems are characterized by solutions consisting of both rapidly and slowly changing components; examples occur in areas such as circuit analysis,

reactor kinetics, and distillation column modeling, where various phenomena happen with widely varying time scales. The rapidly changing (decaying) components are short-lived transients and the slowly changing components eventually settle into a "steady-state" solution. (Sometimes the transients can be removed from the model by pseudo-steady-state assumptions, but not always.) Equation systems exhibiting such behavior are called "stiff". Although the transient components may be of importance during only a very short portion of the overall integration interval, they may limit the maximum step-size that can be used over the entire integration interval.

Although stiffness usually arises for systems of equations, one can observe the stiffness phenomenon in the solution of the scalar equation of the general form

$$\frac{dy}{dx} = f(x,y) = \lambda(y(x) - g(x)) + \frac{dg(x)}{dx} , \qquad (1.64)$$

where λ is a constant (the model equation (1.38) is a special case of (1.64)). The solution is

$$y(x) = g(x) + [y(x_0) - g(x_0)]e^{\lambda x} . \qquad (1.65)$$

If g(x) is slowly varying (eventually settling down to some final value) and Re(λ) is large and negative (implying a rapidly changing transient component that dies out quickly and thereafter contributes little to y(x)), then (1.64) is a stiff problem. For example, the equation

$$\frac{dy}{dx} = f(x,y) = -10^5 y + 10^5 e^{-x} + e^{-x} , \qquad (1.66)$$

has the analytical solution (for $y(x_0) = y(0) = 0$)

$$y(x) = e^{-x} - e^{-10^5 x} . \qquad (1.67)$$

Here, $-e^{-10^5 x}$ is a rapidly decaying transient component superimposed on a slowly changing "steady-state" component, e^{-x}. For $x > 1.4 \times 10^{-4}$

the transient component is smaller than 10^{-6} in magnitude, and has effectively no influence on the solution y(x), as shown in Figure 7.

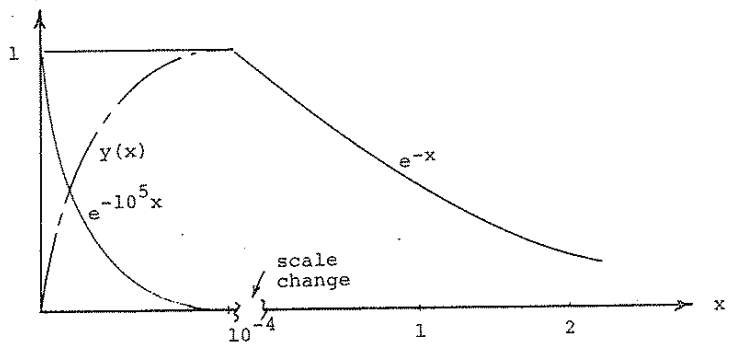

FIGURE 7 Solution of equation (1.66) with y(0) = 0

Nevertheless, if Euler's method (1.9) were used to solve (1.66), the propagation factor from (1.10) would apply, and the corresponding absolute stability criterion of (1.41) would restrict the stepsize to

$$|1 - 10^5 h| < 1 \text{ or } h < 2 \times 10^{-5} . \qquad (1.68)$$

For accurate solution during the transient period one would expect to have to use a very small stepsize (much smaller than 10^{-5}), but once the transient is insignificant, it would seem reasonable to use much larger stepsizes to generate the e^{-x} portion of the solution. Criterion (1.68) shows that this is not the case. If one attempts to use values of h larger than 2×10^{-5}, even after the transient has effectively vanished, unstable propagation of error caused by the transient component will overwhelm the true solution and render the results meaningless.

On the other hand, if Euler's backward difference formula (1.58) were used, the A stability of the method would insure that, regardless

of stepsize, calculations would yield a stable solution that settles into the steady state solution eventually. Of course, if accuracy is required, then very small stepsizes would be essential for the transient interval. Once the transient period is over, however, the stepsize could be increased with assurance that the transient component will continue to decay; the transient component will be very inaccuractely computed at this point, but it will be so insignificant that it won't matter.

If the stiff problem is to be solved with reasonable computational effort, it is essential that the IV algorithm used possess A stability (better-yet, strong A stability), or a very close approximation to it. Thus, for stiff problems, the classical explicit RK methods and the higher-order LMS methods such as the fourth-order Adams predictor-corrector method, simply cannot be used because of their very restricted absolute stability regions.

The semi-implicit RK methods described in the preceding subsections have appropriate stability (and high-order accuracy) for solving stiff systems, but have not been used very extensively to date. This is probably due in part to the difficulty of estimating local truncation errors that is inherent in most RK methods (some of the imbedded forms and Cash's method (12) overcome this problem, however).

The more important reason is that several high-order LMS correctors having near A stability have been developed by Gear (29,30), and incorporated into widely available software packages. Gear's backward difference formula (BDF) of order p has the form

$$y_{i+1} = \sum_{j=0}^{p-1} \alpha_j y_{i-j} + h\beta_{-1} f_{i+1} . \qquad (1.69)$$

Only one estimated derivative, f_{i+1}, is used, and the method is implicit in y_{i+1}. Hence the BDF forms are LMS corrector equations. The first-

order member of the family is Euler's backward formula (1.58). Formulas for orders p = 1 - 6 are:

order p

1 $y_{i+1} = y_i + hf_{i+1}$,

2 $y_{i+1} = \frac{4}{3}y_i - \frac{1}{3}y_{i-1} + \frac{2}{3}h\, f_{i+1}$,

3 $y_{i+1} = \frac{18}{11}y_i - \frac{9}{11}y_{i-1} + \frac{2}{11}y_{i-2} + \frac{6}{11}f_{i+1}$, (1.70)

4 $y_{i+1} = \frac{48}{25}y_i - \frac{36}{25}y_{i-1} + \frac{16}{25}y_{i-2} - \frac{3}{25}y_{i-3} + \frac{12}{25}f_{i+1}$,

5 $y_{i+1} = \frac{300}{137}y_i - \frac{300}{137}y_{i-1} + \frac{200}{137}y_{i-2} - \frac{75}{137}y_{i-3} + \frac{12}{137}y_{i-4} + \frac{60}{137}f_{i+1}$,

6 $y_{i+1} = \frac{360}{147}y_i - \frac{450}{147}y_{i-1} + \frac{400}{147}y_{i-2} - \frac{225}{147}y_{i-3} + \frac{72}{147}y_{i-4} - \frac{10}{147}y_{i-5} + \frac{60}{47}f_{i+1}$

Gear has shown that these correctors have a property that he calls stiff stability defined as follows:

A method is stiffly stable for the model problem (1.38) if in the region [$Re(h\lambda) \leq d$], it is absolutely stable, and in the region [$d < Re(h\lambda) < e$, $|Im(h\lambda)| < \phi$) it is accurate.

Graphically, the stiffly stable region of the complex $h\lambda$ plane is shown in Figure 8. A component of a solution for which λ is large and negative (a rapid transient) can be computed accurately with sufficiently small h, and when no longer of significance will remain stable, even for very large h. The upper bound for the absolutely stable region varies from d = -6.1 to 0 for orders 6 through 2, and the minimum value for ϕ is about 0.5.

Although stiffness can occur in solution of a scalar ODE [see (1.65)], stiff problems normally arise for systems of ODE whose solutions contain components of widely varying time scale. The simplest such system is the uncoupled n-dimensional analog of the model problem (1.58),

$$\frac{dy_j}{dx} = f_j(x,y_j) = \lambda_j y_j = a_{jj} y_j , \qquad (1.71)$$

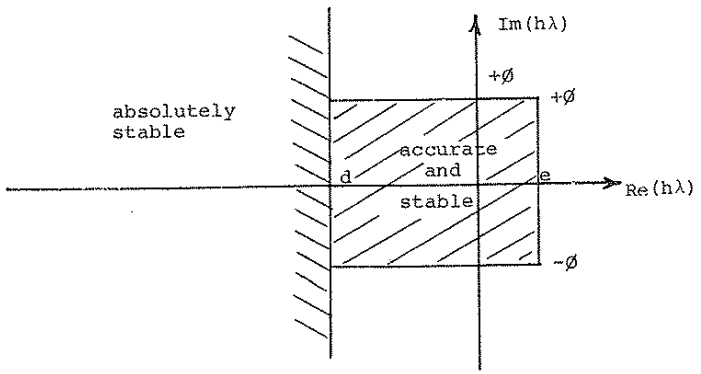

FIGURE 8 Stable and Accurate Regions for the Stiffly-Stable Gear BDF Correctors

which can be written in matrix form as

$$\bar{y}' = \bar{f}(x,\bar{y}) = A\bar{y} . \tag{1.72}$$

Here, $\bar{y}$ is a column vector $(y_1, y_2, \ldots, y_n)^t$, $\bar{y}'$ is a column vector containing the derivatives of $\bar{y}$, and A is an n by n matrix, in this case diag $(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

The matrix A can also be viewed as the Jacobian for the equation system, J, where

$$J_{jk} = \frac{\partial f_j(x,y)}{\partial y_k} \tag{1.73}$$

Since J is in diagonal form, the $a_{jj} = \lambda_j$ are simply the eigenvalues of the Jacobian, also called the eigenvalues of the ODE system.

The solutions of (1.71) are

$$y_j(x) = y_j(0)\exp(\lambda_j x) , \tag{1.74}$$

or

$$\bar{y}(x) = e^{Ax}\bar{y}(0) ,$$

where e^{Ax} is an n by n diagonal matrix.

If (1.71) is solved using Euler's forward formula, the algorithm (analogous to (1.42) for the scalar problem) is

$$\bar{y}_{i+1} = (I + hA)\bar{y}_i , \tag{1.75}$$

Here, I is the nth-order identity matrix and $\bar{y}_i$ and $\bar{y}_{i+1}$ are vectors of the numerical solutions at x_i and x_{i+1}, respectively. The absolute stability criterion corresponding to (1.41) for the scalar problem is

$$-2I < hA < \bar{\bar{0}} \tag{1.76}$$

where $\bar{\bar{0}}$ is the null matrix. Note that (1.76) requires that the stepsize h satisfy

$$0 < h < \min_j 2/|\lambda_j| , \tag{1.77}$$

assuming that all the real parts of the λ_j are negative. Thus the stepsize is limited by the eigenvalue of largest magnitude for the equation system.

We can define a set of time constants, one for each of the exponential solution functions,

$$\tau_j = -1/\lambda_{jR} , \tag{1.78}$$

where λ_{jR} is the real part (assumed to be negative) of eigenvalue λ_j,

$$\lambda_j = -1/\tau_j + i\omega_j . \tag{1.79}$$

The jth exponential in the solution (1.74) can be seen to be the product of oscillatory (if λ_j has an imaginary component) and decaying (from the real part) factors

$$\exp(\lambda_j x) = \exp(i\omega_j x)\exp(-x/\tau_j) . \tag{1.80}$$

In (1.79) and (1.80), i is the imaginary root.

Thus, if one or more of the time constants λ_j is small in comparison with the others, the problem will be a stiff one requiring stepsizes for Euler's method comparable to the smallest time constant. A common measure of stiffness is the stiffness ratio

$$S = \tau_{max}/\tau_{min} , \tag{1.81}$$

where τ_{max} and τ_{min} are the largest and smallest time constants, respectively. If S is near 1, the problem is not stiff, if near 100 midly stiff, if of order 10^4 moderately stiff, and greater than 10^6 very stiff. Hindmarsh and Byrne (31) suggest that a better stiffness measure is

$$S = x_{max}/\tau_{min}, \tag{1.82}$$

where x_{max} is the upper limit of integration (assuming that $x_0 = 0$); in (1.82), S is approximately the total number of steps required for the stable solution of the ODE by Euler's forward formula.

For the backward Euler method (1.58), the solution of (1.72) is

$$\bar{y}_{i+1} = (I - hA)^{-1}\bar{y}_i, \tag{1.83}$$

and, as in the scalar case, the method has A stability; h is unlimited insofar as numerical stability is concerned.

Next, consider the solution of (1.72) where A is not necessarily diagonal, using the backward Euler method of (1.83). The ODE are coupled and A is still the Jacobian for the system. If A is diagonalized by an appropriate similarity transformation U, and a new dependent variable $\bar{z} = U^{-1}\bar{y}$ defined (see (31)), the coupled equations can be uncoupled with the form

$$\bar{z}' = D\bar{z}. \tag{1.84}$$

Here, $D = U^{-1}AU$ has the same eigenvalues as A. The solutions of the coupled set are

$$y_j(x) = \sum_{k=}^{n} z_j(0)u_{jk}\exp(-\lambda_j x). \tag{1.85}$$

Thus the stiffness conditions for the coupled system are exactly the same as for the uncoupled system, and the same stability and stepsize considerations hold.

When the backward Euler method is used, it is necessary to solve the simultaneous equations (1.83), which in this case are linear; a

linear equation solver can be used to find $\bar{y}_{i+1}$.

If the vector form of (1.64),

$$\bar{y}' = A(\bar{y}(x) - \bar{g}(x)) + \frac{d\bar{g}(x)}{dx}, \tag{1.86}$$

is to be solved, letting $\bar{r} = \bar{y}(x) - \bar{g}(x)$ transforms the equations (1.86) to the coupled form of (1.72), which can in turn be uncoupled by the device of the preceding paragraphs. The analog of (1.83) for this problem is

$$\bar{y}_{i+1} = (I - hA)^{-1}(\bar{y}_i + \bar{b}(x)), \tag{1.87}$$

where $\bar{b}(x)$ may depend on x but not on $\bar{y}$. The solution of (1.87) is no more difficult that that of (1.83).

In most problem situations, of course, the ODE are not only coupled but nonlinear as well; i.e., the equations have the form

$$\bar{y}' = \bar{f}(x,\bar{y}). \tag{1.88}$$

The solution of the algebraic equations generated by an implicit algorithm presents some problems, since the equations are nonlinear. An iterative algorithm is required. A typical equation corresponding to the backward Euler method would be

$$y_{j,i+1} = y_{j,i} + hf_j(x_{i+1},\bar{y}_{i+1}). \tag{1.89}$$

One might be tempted to treat (1.89) as a typical corrector and rewrite it in successive substitution form as

$$y_{j,i+1,m+1} = y_{j,i} + hf_j(x_{i+1},\bar{y}_{i+1,m}), \tag{1.90}$$

where m is the iteration counter. Unfortunately, the restrictions on stepsize required for convergence of the iteration (see (1.34)) are so stringent when compared with the essentially unlimited stepsizes allowed by the stability criterion, that (1.90) cannot be employed with success on stiff problems.

Newton-Raphson or quasi-Newton algorithms are normally used to

solve the implicit corrector equations. The Newton-Raphson method leads to the solution of the linear equations

$$\hat{J}(x_{i+1},\bar{y}_{i+1,m})\bar{\delta}_m = -\bar{F}(x_{i+1},\bar{y}_{i+1,m}), \tag{1.91}$$

where m is the iteration counter and

$$F_j(x_{i+1},\bar{y}_{i+1}) = y_{j,i+1} - y_{j,i} - hf_j(x_{i+1},\bar{y}_{i+1}), \tag{1.92}$$

$$\bar{y}_{i+1,m+1} = \bar{y}_{i+1,m} + \bar{\delta}_m. \tag{1.93}$$

$\hat{J}$ is the Jacobian of the system of equations $\bar{F}(x_{i+1},\bar{y}_{i+1}) = \bar{0}$. The iteration of (1.92) is continued until $\bar{\delta}_m$ becomes sufficiently small. It is very important that the elements of F on the right-hand side of (1.91) be computed accurately for each iteration. Fortunately, however, the Jacobian need only be approximate; it is usually not necessary to update the changing elements of $\hat{J}$ with each iteration. In fact, it may be possible to use the same Jacobian for more than one step. This can result in the saving of considerable computing time, since the evaluation and inversion of $\hat{J}$ can be very costly, particularly for large systems. Additionally, it should be noted that the Jacobian of the equation system $\hat{J}$ is related to the Jacobian of the ODE system J by

$$\hat{J} = I - \beta_{-1}J \tag{1.94}$$

for Gear's backward difference formulas.

In many (probably most) large systems of coupled ODE resulting from analysis of physical systems, the elements of $\hat{J}$ have a banded structure, and banded or sparse linear equation solvers can be used. In some cases, the Jacobian may be so diagonally dominant that $\hat{J}$ can be approximated by its diagonal entries only, in which case, $\hat{J}^{-1}$ is simply the diagonal matrix whose entries are the reciprocals of those in $\hat{J}$. Equations (1.91) can be solved by a simple matrix multiplication. The Jacobian elements can be computed from analytical expressions when available, but

are frequently computed numerically by perturbation.

STARTING THE SOLUTION, ADJUSTING THE STEPSIZE

The designer of general-purpose software for solution of the IV problem has a tremendous number of options as to choice of method, strategy for error control, etc. Some of the questions to be answered are the following:

1. Is the program needed for the solution of stiff problems or only nonstiff ones?
2. Which class of methods will be used?
3. Will the method order be fixed or variable?
4. How are the starting values to be determined?
5. How will the stepsize be adjusted to maintain errors within user-specified bounds?

Since existing software for solving ODE is being covered in the paper by Byrne (32), we will touch only briefly on the answers to the above questions.

Although other methods for solution of the IV problem are available (e.g., extrapolation methods in which Romberg-like repeated Richardson's extrapolation procedures are applied to results of a low-order IV algorithm such as Euler's method, to produce results of high accuracy (17)), the choice of method class will usually come down to the RK or LMS methods. If stiff problems are to be handled, then the choice is essentially limited to the semi-implicit RK methods and Gear's backward difference LMS formulas. For the latter, the BDF correctors would normally be used with suitable LMS predictors in a predictor-corrector implementation. For nonstiff problems, explicit RK methods or LMS predictor-corrector methods with limited absolute stability ranges such as Hamming's method or the Adams formulas would be the typical choices

(Gear's formulas can also be used for nonstiff problems, but have larger truncation error terms than the nonstiff LMS methods of the same order).

The next choice would be to decide on a fixed-order or variable-order method. If the choice is for fixed-order, then fourth-order methods have been found to be of acceptable accuracy for most engineering problems, but the end use of the routine would probably dictate the selection. If a variable order is to be allowed, then all orders of a particular class of methods such as the first-to sixth-order Gear's formulas, or the Adam's PC methods of orders one through twelve might be chosen (variable-order RK programs seem to be less common).

As to starting values, the RK methods are, of course, self-staring. The LMS methods are normally not self-starting (except for the first- or second-order ones), so some other technique must be used to establish the values of y_i and f_i needed before the first application. If a fixed-order LMS algorithm has been picked, then an RK method of comparable order would normally be used for the first few steps. If a variable-order method (these usually employ the LMS methods, but variable-order RK programs could be written too) is chosen, then the lowest-order formula is used with very tiny stepsize (to insure accuracy) to generate the first needed values; gradually the order is increased until sufficient information is present to continue with essentially any order algorithm.

The user will normally want to specify accuracy criteria for his results (relative or absolute bounds for the global error), so the routine must be able to estimate the local error being committed using the approaches outlined. If the method is stable for the ODE being solved, it is usually assumed that the total error over the entire integration interval is approximately the sum of the local truncation errors, so a

conservative local error per unit step can be established in advance. A stepsize adjustment strategy must then be incorporated into the program. For a fixed-order method, this normally involves decreasing or increasing the current step length by some factor (fixed or variable depending on the apparent rate of change of the error); often, the step length is simply doubled or halved (sometimes repeatedly until the error criteria are satisfied). With RK methods, the stepsize adjustment is quite simple, since all are one-step methods. The LMS methods present some problems, since the formulations of the equations are usually in terms of fixed h. Several approaches are possible. First, the methods can be written for variable stepsize (this turns out to be quite cumbersome for the higher-order methods). Another is to generate any essential values (when the stepsize is halved, for example, additional y_i and f_i values are needed) using a one-step RK method of comparable order. Still another, would be to fit interpolating polynomials to past values of the y_i and f_i and evaluate the polynomials at the necessary new argument locations. Nordsieck (33) has proposed an alternative approach to the storage of historical information (old values of the y_i and f_i). He uses the values to generate what amounts to a power series expansion of the solution function at the current x; essentially, higher-order scaled derivatives are approximated at x_i such that, for any h, the appropriate y and f values can be regenerated. The scaled derivatives are updated at each step, and any desired change for the next step can be accommodated.

Finally, there is the possibility of varying both the stepsize and the method order. The object is to reduce the total number of function evaluations needed to achieve integration of the ODE over a fixed interval, subject to limitations on the maximum (and minimum) tolerable errors. The logic for these step and order changes is reasonably com-

plex, and to a considerable extent based on heuristics (and varies from program to program). The most widely used of the general-purpose ODE solvers, written by Byrne and Hindmarsh at Livermore Laboratories (34) use variable-step, variable-order approaches (and in addition incorporate both a stiff and nonstiff option). Presumably, it is possible to write programs that not only change stepsize and order within a method class, but that shift from class to class as well; for example, changing automatically from nonstiff to stiff methods and vice versa, depending on the current stiffness of the equations (in nonlinear ODE systems, the stiffness changes in time, since the Jacobian and hence its eigenvalues vary from step to step).

Nearly every paper and text referenced in this section contain computed results for solution of one or more ODE problems. Frequently, results are compared with those produced by other programs for other IV techniques. Certain problems have become "classical" in that several authors have used them as benchmarks for testing purposes. Unfortunately, comparison of the results of two different programs using two different methods is fraught with some peril. Differences in efficiency, for example, may be less a function of the method used than of the particular step-changing strategy employed or the ability of the programmers involved (some write excellent code, others sloppy code, etc.). Nevertheless, there are several sources of such comparisons that may be of interest, in particular in (14), (17), (35), and (36).

BOUNDARY-VALUE METHODS FOR THE SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

As is indicated in the introduction to the previous section, solution methods for boundary-value (BV) problems are, in general, less

satisfactory and computationally more expensive than are the well-developed and reliable methods for the initial value (IV) problem described in the previous sections. Unlike the IV problem that typically has a unique solution, the BV problem may have many solutions(or none). The two-point BV problem is the one most commonly encountered in chemical engineering, and in its simplest form consists of a single second-order ODE with two associated conditions:

$$F(x, y, y', y'') = 0, \qquad (2.1)$$

$$y(a) = \alpha, \; y(b) = \beta.$$

The treatment here will be brief, partly because there is a dearth of good techniques and partly because most of the methods involve adaptations of solution algorithms already described elsewhere in the paper. Five approaches will be discussed: (1) shooting methods, (2) parallel shooting methods, (3) finite-difference methods, (4) quasilinearization, and (5) collocation.

SHOOTING METHODS

In the shooting methods, the BV problem is solved repeatedly using IV methods. The ODE (one equation of order n or possibly a system of equations of mixed order) are usually decomposed into a system of first-order equations, suitable for solution by an IV software package. Certain of the boundary conditions will be specified at the "initial" point, say x_0. Any missing initial conditions for the dependent variables are guessed, and the ODE are integrated using an IV program. The computed values y_k at arguments x_k for which the remaining boundary conditions are specified are then compared. New guesses are made for the missing conditions, the IV problem is solved again for these new conditions, and a new comparison of the remaining boundary conditions and their computed counterparts is made. This process is repeated until the guessed

missing initial conditions produce results at the remaining boundary points that are suitably accurate. The approach is called "shooting" for obvious reasons.

For example, consider equation (2.1) with $y(a) = y(x_0) = \alpha$, $y(b) = y(x_n) = \beta$ specified. In order to use an IV method to solve this second-order problem (probably rewritten as two first-order equations), both $y(x_0)$ and $y'(x_0)$ must be specified. Thus, $y'(x_0)$ must be guessed. Let $y_n(c)$ be the computed value of y_n at x_n, given $y'(x_0) = c$. If two different solutions are found for $c = c_1$ and $c = c_2$ the results might appear as shown in Figure 9.

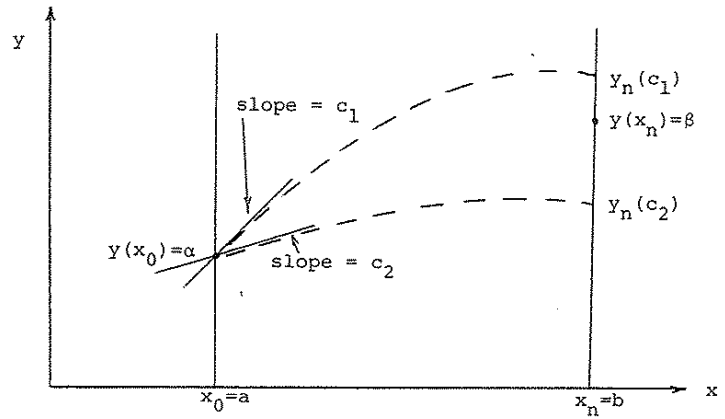

FIGURE 9 Shooting method solution of a two-point boundary-value problem

If equation (2.1) were linear, then the "true" solution could be found as the linear combination of the computed solutions,

$$y_i = y_i(c_1) + \mu y_i(c_2) \,, \qquad (2.2)$$

with

$$\beta = y_n(c_1) + \mu y_n(c_2) \,,$$

but in general (2.1) will not be linear, and (2.2) is not applicable.

The usual procedure is to solve the nonlinear equation

$$\delta = \beta - y_n(c) = 0 \qquad (2.3)$$

for c using some root-finding technique, such as Newton's method, a secant method, or the half-interval method.

When there is more than one missing initial condition, a system of nonlinear equations comparable to (2.3) will be generated and an appropriate method (e.g., Newton-Raphson, multi-dimensional secant algorithms) can be used to solve iteratively for the missing conditions. Each iteration, of course, requires a new solution of the IV problem. Alternatively, one can define a nonnegative objective function such as

$$\sum_{i=1}^{k} \delta_i^2 \,, \qquad (2.4)$$

where δ_i is the discrepancy between the ith non-initial boundary condition and its computed counterpart, given a set of values for the missing conditions. Some optimization algorithm, such as Marquardt's method, can then be used to minimize (2.4).

PARALLEL SHOOTING

When the solution is found to be quite sensitive to the guessed values of the missing initial conditions, the shooting method of the preceding section may fail. An extension of the method called parallel shooting is often found to produce good results. To illustrate, consider the second-order ODE of (2.1). The interval ($x_0 = a$, $x_n = b$) is partitioned into n subintervals, usually but not necessarily of uniform length. A new variable t is introduced, such that on each subinterval $0 \le t \le 1$. Then y(x) and y'(x) are replaced by a 2n-element vector

$$\bar{y}(t) = [y_1(t), y_2(t), \ldots, y_n(t), y_1'(t), y_2'(t), \ldots, y_n'(t)]^t \,, \qquad (2.5)$$

with

$$y_i(t) = y[(x_{i-1}) + t(x_i - x_{i-1})] \,. \qquad (2.6)$$

Thus $y_i(t)$ describes the solution in subinterval i.

The second order ODE of (2.1) is then rewritten as a first-order system of 2n ODE

$$\bar{y}'(t) = \bar{f}(t,\bar{y}) , \tag{2.7}$$

and the solutions $y_i(t)$ and $y_i'(t)$ are required to match at the interior subinterval junction points, i=1,2,...,n-1:

$$y_{i+1}(0) = y_i(1) , \tag{2.8}$$

$$\frac{y'_{i+1}(0)}{x_{i+1}-x_i} = \frac{y'_i(1)}{x_i-x_{i-1}} . \tag{2.9}$$

Conditions (2.8) and (2.9) introduce 2(n-1) new boundary conditions, in addition to the two specified in the original problem.

Guesses are made for the missing 2n-1 elements of $\bar{y}(0)[y_0(0) = \alpha]$, and the shooting method of the previous subsection is applied to the system of 2n simultaneous ODE of (2.7); any appropriate IV routine can be used to carry out the integrations with t as the independent variable. Then the 2n-1 simultaneous (nonlinear) equations

$$\begin{aligned} g_i(\bar{y}(0)) &= y_{i+1}(0) - y_i(1) = 0, \quad i=1,2,\ldots,n , \\ q_i(\bar{y}(0)) &= y'_{i+1}(0)/(x_{i+1}-x_i) - y'_i(1)/(x_i-x_{i-1}) = 0 , \\ &\qquad i=1,2,\ldots,n-1 , \\ q_n(\bar{y}(0)) &= y_n(1) - \beta , \end{aligned} \tag{2.10}$$

must be solved by some nonlinear equation solving or minimization techniques for the next set of missing initial conditions. The final equation in (2.10) insures that the originally given boundary condition at x_n is satisfied.

The solution technique can be refined by an imbedding approach in which n is changed from 2 to 4 to 8, etc. The missing conditions at the initial point and at the central point for n = 4 are taken as the final

values at the initial and central points, respectively, for n = 2, etc.; the other missing conditions are usually found by interpolation.

Scott and Watts (37) discuss different approaches to the solution of ODE by shooting and parallel shooting techniques, including the solution of equations (2.10) by quasi-Newton methods that use Broyden's rank-one updating procedure for approximating the inverse of the Jacobian. Na (38) and Keller (39) show how elements of the Jacobian can be found by integrating a set of auxiliary ODE simultaneously with the ODE of (2.7).

FINITE DIFFERENCE METHODS

One of the most straightforward approaches to solution of BV problems is to divide the integration interval into n uniform subintervals, and then to replace derivatives appearing in the ODE by their finite-difference approximations, as described in the next section of the paper (equations (3.2 through 3.5), or their higher-order counterparts). If (2.1) is linear, such substitutions (with the boundary conditions incorporated as knowns, y_0 and y_n) lead to a system of n-1 simultaneous linear equations in the unknowns $y_1, y_2, \ldots, y_{n-1}$ at the interior grid-points (joints of the subintervals). The system has a tridiagonal structure (3.15) and can be solved easily by the straightforward algorithm of (3.16) and (3.17).

Unfortunately, when the ODE is nonlinear, the difference equations are also nonlinear, and one difficult problem (the BV problem) has been replaced by another (solution of a possibly very large system of nonlinear equations). The finite-difference methods are usually not used in this fashion for nonlinear ODE, as better methods are available.

QUASILINEARIZATION

Quasilinearization is a method originally advocated by Bellman (40).

Although somewhat tedious computationally, the approach is conceptually simple. Consider the solution of n first-order ODE

$$\frac{\bar{dy}}{dx} = \bar{f}(x,\bar{y}) , \tag{2.11}$$

with n boundary conditions specified at two points x_0 and x_m, arranged as follows:

$$\begin{aligned} y_{j,0} &= y_j(x_0), \quad j=1,2,\ldots,q \\ y_{j,m} &= y_j(x_m), \quad j=q+1,\ldots,n , \end{aligned} \tag{2.12}$$

with $1 \leq q < n$ (this ordering simplifies the notation later in a detailed development, but isn't crucial).

Let $z_j(x)$ be an approximation of the true solution $y_j(x)$ that satisfies the boundary conditions. Then the right-hand sides of (2.11) are expanded in Taylor's series about these approximate solutions $\bar{z}(x) = [z_1(x), z_2(x),\ldots,z_n(x)]^t$ with only the linear terms retained to yield

$$\frac{dy_j}{dx} = f_j(x,\bar{y}(x)) \doteq f_j(x,\bar{z}(x)) + \sum_{k=1}^{n} \frac{\partial f_j}{\partial z_k}(y_k(x) - z_k(x)) , \tag{2.13}$$
$$j=1,2,\ldots,n ,$$

Since the $z_j(x)$ are known (assumed) functions of x, (2.13) is a system of linear ODE with variable coefficients which can then be solved using the finite difference approach described later.

Usually, n new variables $\varepsilon_j(x)$ are introduced, where

$$\varepsilon_j(x) = y_j(x) - z_j(x) \tag{2.14}$$

is simply the difference (error) between the true and assumed solutions for the jth variable. Introducing (2.14) into (2.13) yields

$$\frac{d\varepsilon_j}{dx} \doteq f_j(x,\bar{z}(x)) - \frac{dz_j}{dx} + \sum_{k=1}^{n} \frac{\partial f_j}{\partial z_k}\varepsilon_k(x) , \tag{2.15}$$
$$j=1,2,\ldots,n ,$$

with boundary conditions (2.12) transformed to

$$\begin{aligned} \varepsilon_j(x_0) &= 0, \quad j=1,2,\ldots,q , \\ \varepsilon_j(x_0) &= 0, \quad j=q+1,\ldots,n . \end{aligned} \tag{2.16}$$

Now the interval is partitioned into m subintervals, usually of equal length $h = (x_m - x_0)/m$, and substitution of finite-difference approximations for the total derivatives leads to a system of nm linear algebraic equations,

$$\begin{aligned} &\varepsilon_{j,i-1} + \frac{h}{2}\sum_{k=1}^{n}\left(\frac{\partial f_j}{\partial z_k}\right)\varepsilon_{k,i-1} - \varepsilon_{ji} + \frac{h}{2}\sum_{k=1}^{n}\left(\frac{\partial f_j}{\partial z_k}\right)\varepsilon_{ki} \\ &\quad = -h\left(\frac{f_{j,i-1} + f_{ji}}{2} - \frac{z_{ji} - z_{j,i-1}}{h}\right) ; \qquad \begin{matrix} j=1,2,\ldots,n \\ i=1,2,\ldots,m \end{matrix} \end{aligned} \tag{2.17}$$

where

$$\begin{aligned} \varepsilon_{ji} &= \varepsilon_j(x_i) , \\ f_{ji} &= f_j(x_i,\bar{z}(x_i)), \\ z_{ji} &= z_j(x_i) , \\ \frac{\partial f_j}{\partial z_k} &= \frac{\partial f_j}{\partial z_k}_{(x_{i-1},x_i)} \end{aligned} \tag{2.18}$$

With the boundary conditions (2.16) specified, the equations of (2.17) can be solved using LU decomposition or a sparse matrix equation solver to yield the unknowns:

$$\begin{aligned} &\varepsilon_{j0}, \quad j=q+1,q+2,\ldots,n , \\ &\varepsilon_{jm}, \quad j=1,2,\ldots,q , \\ &\varepsilon_{ji}, \quad j=1,2,\ldots,n; \quad i=1,2,\ldots,m-1 . \end{aligned} \tag{2.19}$$

The partial derivatives in (2.17) should be found analytically, if possible, and evaluated at mid-interval, in which case any z values needed are taken as the average of the values at the endpoints of the pertinent interval.

Once (2.17) has been solved for the unknowns (2.19), (2.14) can be used to update the estimates of the solution values $\bar{z}(x)$. The process is repeated until all of the solutions (18) become acceptably small. Two examples of interest to chemical engineers that use the quasilinearization approach (where shooting methods failed to converge) are shown in (41) and (42). Scott and Watts (37), who prefer shooting methods when they work, have noted that quasilinearization appears to have a larger domain of convergence for many problems than do the shooting methods.

COLLOCATION

Collocation is a strategy for solving boundary-value problems in both PDE and ODE. For simplicity, we treat the one-dimensional (ODE) case. The technique has many similarities to the finite-element method described in the final section of the paper, and the notation from that section will be used here. The solution of the differential equation

$$L\,u = f, \tag{2.20}$$

where L is a linear differential operator (i.e., $L(\alpha y + \beta v) = \alpha Ly + \beta Lv$) is approximated by a trial function v that is a linear combination of n linearly independent basis functions, $\psi_1,\ldots,\psi_n$,

$$u \doteq v = \sum_{j=1}^{n} c_j \psi_j . \tag{2.21}$$

Here, the basis functions are usually simple functions (e.g., polynomials) modified by some function that insures satisfaction of the boundary conditions. Collocation ("setting or placing together") then consists of determining the coefficients c_j such that

$$Lv = L\left(\sum_{j=1}^{n} c_j \psi_j\right) = f \tag{2.22}$$

is satisfied exactly at n "collocation points", x_i, i=1,2,...,n, yet to be selected (they might be evenly spaced between the two boundary points,

for example). Then (2.22) leads to a system of n simultaneous linear equations whose coefficient matrix contains elements

$$a_{ij} = L\psi_j(x_i) . \tag{2.23}$$

Orthogonal collocation is a special case of collocation in which the basis functions consist of members of an orthogonal polynomial family as the principal part, and the collocation points are the roots of the nth-order member of the family. If the polynomials are selected properly, the result is identical to that obtained by the Galerkin method (4.4), and hence to the variational equivalent (the final section of the paper), without the need to perform the tedious integrations.

An example should help in understanding the approach. Let the equations and associated boundary conditions be:

$$Lu = \frac{d^2u}{dx^2} + au = f, \tag{2.24}$$

$$x = 0: \quad \frac{du}{dx} = 0,$$

$$x = 1: \quad u = 0.$$

The first and second conditions are called natural and essential conditions, respectively. Anticipating that the solution u is likely to be an even function of x, let the basis functions be

$$\psi_j = (1 - x^2)P_{j-1}(x^2) . \tag{2.25}$$

Note that the selected functions satisfy both boundary conditions. As shown later, the variational or Galerkin formulations automatically satisfy natural boundary conditions, so the assumed functions ψ_j would need have satisfied only the essential condition u(1) = 0. The trial function will then have the form

$$v = (1 - x^2)\sum_{j=1}^{n} c_j P_{j-1}(x^2), \tag{2.26}$$

which is a polynomial of degree n in the variable x^2 (i.e., a polynomial of degree 2n in x; the number of collocation points must be n).

Now, following Galerkin (4.4), we require

$$I_j = \int_0^1 \psi_j(Lv - f)dx = 0, \quad j=1,2,\ldots n . \tag{2.27}$$

Substituting, (2.26) into (2.27) yields

$$I_j = \int_0^1 (1-x^2)P_{j-1}(x^2)[Lu(x^2) - f(x^2)]dx = 0 , \quad j=1,2,\ldots,n . \tag{2.28}$$

Now, recall that numerical integration formulas of the form

$$\int_a^b w(x)F(x)dx \doteq \sum_{i=1}^{n} w_i F(x_i) \tag{2.29}$$

can be made exact for F(x) a polynomial of degree 2n-1 or less if the base points for the quadrature formula are the zeros of an nth degree orthogonal polynomial $p_n(x)$ satisfying

$$\int_a^b w(x)p_n(x)p_m(x)dx = 0, \quad m < n . \tag{2.30}$$

(This is the basis of the Gaussian quadrature formulas, as described in (19)).

Since u is a polynomial of degree n in x^2 (and assuming that $f(x^2)$ is a polynomial of degree no greater than n in x^2), the integral of (2.28) has the form of (2.29) where $w(x) = (1 - x^2)$ and $F(x) = P_{2n-1}(x^2)$. Hence Galerkin's integral of (2.27) can be evaluated from (2.29) as

$$I_j = \sum_{i=1}^{n} w_i P_{j-1}(x_i^2)[Lu(x_i^2) - f(x_i^2)], \tag{2.31}$$

where the x_i are the zeros of $P_n(x^2)$. But if we collocate, the term in brackets is made to be exactly zero. Hence, if the collocation points are chosen to be the n roots of $P_n(x^2)$, the Galerkin conditions will be

satisfied exactly.

In the general case, one must construct the set of orthogonal polynomials for each problem, depending on the integration interval and the weighting function involved. Fortunately, for many commonly occurring combinations of interval and weighting function, the polynomials and their roots are readily available in tabular form (43).

Now, to return to the example problem, we see that the family of polynomials needed should satisfy

$$\int_0^1 (1 - x^2)P_n(x^2)P_m(x^2)dx = 0, \quad m < n . \tag{2.32}$$

The following polynomials and roots are the ones needed:

order	polynomial	roots
0	$P_0(x^2) = 1$	--------
1	$P_1(x^2) = 1 - 5x^2$	0.447214
2	$P_2(x^2) = 1 - 14x^2 + 21x^4$	0.285232 0.765055
3	$P_3(x^2) = 1 - 27x^2 + 99x^4 - 85.8x^6$	0.209299 0.591700 0.871740

For a specific example of a problem having the form of (2.24) consider the ODE

$$\frac{d^2u}{dx^2} - 4u = 4x^2 - 2 , \tag{2.33}$$

with conditions du/dx = 0 at x = 0 and u = 0 at x = 1.

One-term trial function

$$u = c_1\psi_1 = c_1(1-x^2)P_0(x^2) = c_1(1-x^2) \tag{2.34}$$

$$Lu - f = -2c_1 - 4c_1(1-x^2) - 4x^2 + 2 = 0 \tag{2.35}$$

The one collocation point is the root of $P_1(x^2)$, i.e., $x_1 = 0.447214$, which, when substituted into (2.33) leads to $c_1 = 0.23077$. Thus the

one-term trial function is

$$u = 0.23077(1 - x^2). \tag{2.36}$$

Two-term trial function

$$u = c_1(1-x^2)P_0(x^2) + c_2(1-x^2)P_1(x^2), \tag{2.37}$$

$$= (1-x^2)[c_1 + c_2(1-5x^2)],$$

$$Lu - f = (6-4x^2)c_1 + (1-5x^2)(16-4x^2)c_2 + 4x^2 - 2. \tag{2.38}$$

The two collocation points are the roots of $P_2(x^2)$, i.e., $x_1 = 0.285232$, $x_2 = 0.765055$, yielding (from (2.38)) the two simultaneous linear equations:

$$\begin{aligned} 5.67457\ c_1 + 9.298368c_2 &= 1.674571, \\ 3.658763c_1 - 26.314233c_2 &= -0.341237. \end{aligned} \tag{2.39}$$

The solutions are $c_1 = 0.22304$, $c_2 = 0.04398$, yielding the two-term trial solution

$$\begin{aligned} u &= (1-x^2)[(0.22304 + 0.04398(1-5x^2)], \\ &= (1-x^2)(0.26702 - 0.21990x^2). \end{aligned} \tag{2.40}$$

A comparison of the one-term and two-term solutions with the exact solution shows quite good agreement:

x	u (exact)	u (othogonal collocation) 1 term	2 term
0.00	0.2658	0.2308	0.2670
0.25	0.2372	0.2163	0.2374
0.50	0.1602	0.1731	0.1590
0.75	0.0628	0.1010	0.0627
.00	0.0000	0.0000	0.0000

FINITE-DIFFERENCE METHODS FOR THE SOLUTION OF PARTIAL DIFFERENTIAL EQUATIONS

INTRODUCTION

Partial differential equations (PDEs) govern a very wide range of phenomena of engineering and scientific importance, and here we present a number of proven methods for their solution. The two principal numerical techniques are the well-established finite-difference method (FDM) and the more recent finite-element method (FEM), to be discussed in the following sections. In addition, a few special-purpose techniques will also be mentioned, including the method of characteristics and the method of lines. The present section will lean heavily on material from our book (Carnahan, Luther, and Wilkes (19)), to which the reader is referred for additional details.

Linear PDEs of the second order may be classified, if the equation has been reduced by a suitable transformation of the independent variables, to the form

$$\sum_{i=1}^{n} a_i \frac{\partial^2 u}{\partial x_i^2} + \sum_{i=1}^{n} b_i \frac{\partial u}{\partial x_i} + cu + d = 0,$$

in which the coefficients a_i may be 1, -1, or zero. Here, u is the dependent variable and the x_i are the independent variables. Frequently, one of the x_i will be time t, and the remainder will be one or more of the distance coordinates x, y, and z. The following are the main possibilities of interest:

(1) If all the a_i are nonzero and have the same sign, the PDE is of elliptic type.

(2) If all the a_i are nonzero and have, with one exception, the same sign, the PDE is of hyperbolic type.

(3) If one a_i is zero (a_k, for instance) and the remaining a_i