LA-8927-MS

C.3

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36.

[illegible] Methods for [illegible] Differential Equations

Los Alamos

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

This work was supported by the US Department of Energy, Office of Basic Energy Sciences, Department of Applied Mathematics.

LA-8927-MS

UC-32
Issued: January 1982

Numerical Methods for Nonlinear Differential Equations

James M. Hyman

Los Alamos
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

NUMERICAL METHODS FOR NONLINEAR DIFFERENTIAL EQUATIONS

by

James M. Hyman

ABSTRACT

New and better methods for the numerical solution of partial differential equations are being developed at an ever-increasing rate. In this report, directed to scientists trained in mathematics but not necessarily in numerical analysis, we try to unify and simplify the underlying crucial points in this development. Most of the new methods can be understood and classified according to how space, time, and boundary conditions are discretized and by how nonlinear algebraic equations that arise in the solution process are solved. We will discuss each point and present numerical examples showing how a simple linear analysis can fail.

INTRODUCTION

A numerical algorithm for the solution of nonlinear partial differential equations (PDEs) can be a highly complicated and problem-dependent process. A method developed for a particular test problem may not work for similar problems. Methods that work well in one space dimension may not be easily extended to two or three dimensions. Linear analysis can rarely ensure accuracy with nonlinear methods or highly nonlinear equations.

Each year significant and powerful algorithms are discovered, but most of these methods have a similar underlying structure. To better predict when a method, which is almost always developed for relatively simple test problems, will extend to more complicated situations, we must understand this underlying structure.

First, we will simplify the structure of these methods to study the general flow of the algorithms, understand their similar patterns of interconnections, and unify them in a single theory. In this synthesis, new properties and common features among seemingly different methods sometimes emerge that were not evident when analyzing a specific method for a specific set of equations.

We do not imply that methods tailored to specific equations should be abandoned. Many excellent methods have come from specialized analysis of a specific set of equations. But, by understanding the general patterns found in all the methods, we may gain a better view of how and why the algorithms work as they do.

The prototype system of PDEs studied here can be written as

$$u_t = f(x,t,u), \quad u(x,o) = u_o \ , \tag{1}$$

where the solution $u(x,t)$ lies in some function space, x is in some domain Ω, and f is a nonlinear differential operator. We use the notation u_t and u_x to represent partial differentiation with respect to time and space. On the boundary of Ω, the solution is constrained to satisfy the boundary condition

$$b(x,t,u(x,t)) = 0, \; x \in \partial\Omega \quad , \tag{2}$$

where b is a nonlinear spatial differential operator.

A discrete numerical method approximates u by an element U in some finite dimensional space whose components are the values of u at a discrete set of mesh points. The differential operators f and b are replaced by discrete operations F and B operating on U.

The discretized approximation to Eqs. (1) and (2) is a constrained system of ordinary differential equations (ODEs),

$$U_t = F(U), \; B(U) = 0 \quad , \tag{3}$$

which are then integrated numerically.

Evaluating $F(U)$ and integrating Eq. (3) often requires solving large sparse systems of algebraic equations. The methods used to solve these equations often determine the success or failure of any numerical approach.

This report is organized so that the crucial choices for methods to discretize space, boundary conditions, time, and methods to solve the algebraic systems are analyzed independently. The numerical examples are in only one space dimension for simplicity. We avoid taking undue advantage of this or any linearity that may be present so that the general conclusions may be extended to more complicated problems in higher dimensions.

SPACE DISCRETIZATION

The numerical approximation of the spatial derivatives and the distribution of the mesh points determine how well the spatial operator $f(u)$ and the solution u will be approximated. We describe some typical methods to approximate spatial derivatives and then we describe how the errors in a calculation are related to the order of accuracy of the method.

Often, important properties of the solution behavior originate at the boundary and the numerical differentiation procedure must take the boundary conditions into account. For simplicity, we will not incorporate the influence of the boundary conditions into the discrete operator until the next section.

Numerical Differentiation

The guiding principle in choosing a numerical method to approximate the spatial operator is that the resulting discrete model should retain as closely as possible all the crucial properties of the original differential operator. For instance, for a hyperbolic PDE, the operator f is antisymmetric, so we try to approximate f by an antisymmetric discrete operator F. For a parabolic PDE when f is dissipative, we approximate f with a dissipative discrete operator. If f is in conservation form, we also choose a conservation form of F.

All spatial differentiation methods we describe follow the same algorithmic flow. At time t during a calculation, we are given the approximate solution vector U at a discrete set of mesh points X and must generate a numerical approximation F(U) of f(u) at these mesh points. When f(u) is a nonlinear spatial operator, it will have terms such as $g(u,x,t)_x$ or $[s(u,x)\ g(u,x,t)_x]_x$. First, all nonlinear functions are evaluated to generate, say, the vectors G and S. Next, G, X, and S are input for a black-box-type subroutine package in the computer library. This subroutine then returns the approximate vectors G_x and $(SG_x)_x$ of derivatives at the mesh points as output.

Thus, the spatial differentiation is totally divorced from the nonlinearities of the PDE. This modularity also reduces the redundancy of programming the same approximation to the spatial derivatives each time they appear in an equation. These differentiation routines are debugged and optimized for a particular machine only once--with no specific PDE in mind.

The numerical differentiation methods we describe fall into three categories: finite difference methods, implicit methods, and transformation methods. The simplest and oldest general technique, the finite difference method, is described first.

Finite Differences

In a finite difference approximation, the function g is expanded at mesh points near x_i in a Taylor series about x_i. These expansions are added and subtracted to give

$$\frac{\partial g}{\partial x} \cong G_x(x_i) = \frac{G_{i+1} - G_{i-1}}{2\Delta x} - \frac{\Delta x^2}{6} g_{xxx}(x_i) + O(\Delta x^4) \tag{4}$$

$$\frac{\partial g}{\partial x} \cong G_x(x_i) = \frac{-G_{i+2} + 8G_{i+1} - 8G_{i-1} + G_{i-2}}{12\Delta x} + O(\Delta x^4) \tag{5}$$

$$\frac{\partial}{\partial x} s \frac{\partial g}{\partial x} \cong \frac{S_{i+\frac{1}{2}}(G_{i+1} - G_i) - S_{i-\frac{1}{2}}(G_i - G_{i-1})}{\Delta x^2} \tag{6}$$

where $\Delta x = \frac{1}{2}(x_{i+1} - x_{i-1})$ and

$$S_{i+\frac{1}{2}} = 2S(x_i)S(x_{i+1})/(S(x_i) + S(x_{i+1})) \quad . \tag{7}$$

Here $O(\Delta x^j)$ stands for a quantity bounded by some constant times Δx^j as Δx goes to zero. The formula for $(sg_x)_x$ uses the harmonic rather than the arithmetic average for $S_{i+\frac{1}{2}}$ to ensure flux continuity in the solution [1]. This crucial property of the differential operator should be retained in the discrete approximation.

The vector G_x of derivative values can be written as $G_x = DG$. Here, D is a banded sparse matrix.

Implicit Methods

The implicit or Padé methods are a generalization of the finite difference methods. Equation (4) can be rewritten as

$$(1 + \frac{\Delta x^2}{6} \frac{\partial^2}{\partial x^2}) g_x(x_i) = \frac{G_{i+1} - G_{i-1}}{2\Delta x} + O(\Delta x^4) \quad . \tag{8}$$

Approximating the second derivative operation with Eq. (6) and using matrix notation, we have

$$D_2 G_x = D_1 G + O(\Delta x^4) \quad . \tag{9}$$

Here, D_2 and D_1 are tridiagonal matrices generated by Eqs. (4) and (6). A similar procedure can be used to approximate g_{xx}.

Often, time discretization methods require solving implicitly defined equations such as Eq. (9). In special cases the implicitly defined space and time systems can be solved simultaneously leading to an exceptionally high-order efficient method. These are called operator compact implicit methods [2].

Transformation Methods

In the transformation or pseudospectral method, G is first mapped by a transformation of the form

$$g = TG = \sum_{j=1}^{m} a_j \phi_j(x) \tag{10}$$

into nondiscrete function space. The basis functions $\phi_i(x)$ and the transformation are chosen so that T and T^{-1} are fast ($\leq$ m log m operations), and so that differentiation D is simple in the transform space. The derivative approximation can then be written as

$$G_x = T^{-1} DTG \quad . \tag{11}$$

Some common transforms are based on the fast Fourier transform where the ϕ_i are trigonometric functions, Tchebyshev, or Legendre polynomials. Choosing the ϕ_i as piecewise polynomials with compact support, such as the B splines, is another good choice. Often the transformation is chosen to incorporate some crucial property such as the periodicity or symmetry of g into the calculation. This can greatly improve the accuracy of G_x.

Grid Resolution

The accuracy of the numerical solution is largely determined by how well F(U) approximates f(u) and how well U resolves the solution u. These errors depend on the accuracy of the derivative approximation and the algorithm used to locally refine the mesh. Because the accuracy of the method strongly influences the grid needed to resolve the solution, we first describe these relationships.

Accuracy

Discretizing with a highly accurate difference scheme keeps the computer time and storage to a reasonable level in multidimensional calculations. The analysis is linear, but in practice, qualitative results usually hold for non-linear problems. Usually, the best we can do for nonlinear equations is to seek out basic relationships for linear equations that will be stable to small perturbations. These are the results that are most often retained.

The truncation error in (3) is the amount by which the mesh point values of the true solution to the differential equation (1) fail to satisfy the difference equation (3). These errors are typically $O(\Delta x^j)$ in size and j is called the order of the method.

For any fixed accuracy criteria, the number of mesh points M_j needed in a jth-order linear calculation is related to the number of mesh points needed by other methods according to the relationship

$$M_1 = C_2 M_2^2 = C_4 M_4^4 = C_6 M_6^6 \quad . \tag{12}$$

For the periodic unidirectional wave equation $u_t = vu_x$, the phase error introduced will be the same using second-, fourth-, or sixth-order differences if the number of mesh points in the calculations satisfy [3]

$$M_2 \cong 0.36\, M_4^2 \cong 0.12\, M_6^3 \quad . \tag{13}$$

Table 1 compares the number of points per wavelength necessary to obtain a given phase error e in the kth Fourier mode of the periodic solution to $u_t = vu_x$ at time t using second-, fourth-, and sixth-order spatial centered differences in (3). In a calculation where the solution contains many different frequencies, the high modes (2-5 points per wavelength) are approximated equally poorly with all the methods. The middle modes (6-16 points per wavelength) are computed much more accurately with the fourth- and sixth-order differences than with the second-order method. The sixth-order differences are more accurate for the lower modes than either second- or fourth-order differences, but this gain is often lost because of errors introduced in the approximation of the boundary conditions.

2nd order M_2	4th order M_4	6th order M_6	Accuracy $e/(vkt)$
4	4	3	2.6
8	5	4	0.65
16	7	5	0.16
32	10	7	0.04
64	14	8	0.01
128	19	10	0.0025
256	27	13	0.0006

Table 1. Points per wavelength for second-, fourth-, and sixth-order differences to have the same accuracy.

The relationship of the accuracies of the different methods compared to the number of points per wavelength is even more impressive in higher dimensions. In two space dimensions, the number in Table 1 should be squared, and in three dimensions, cubed.

The corresponding relationship for the damping error in $U_t = U_{xx}$ for second- vs fourth-order finite differences is [3]

$$M_2 = 0.44\, M_4^2 \quad . \tag{14}$$

The qualitative results from this linear analysis hold true for many nonlinear equations, as shown in Fig. 1 where the density is plotted into solutions of the Euler equations of gas dynamics for a Riemann problem. (A complete description of these calculations is in Ref. 4.) Second-order finite differences were used in Fig. 1a and fourth-order in Fig. 1b, otherwise they are the same. Note that the higher order differences resolve the solution better.

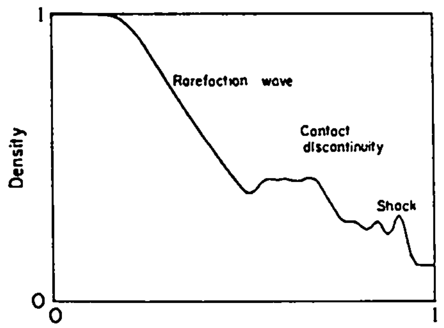

Figure 1a
Second-order

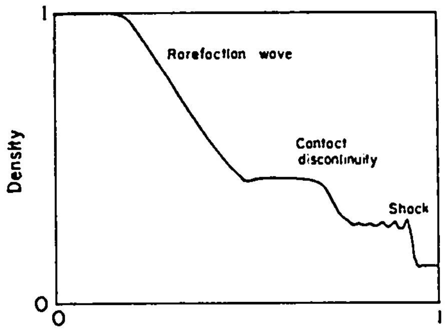

Figure 1b
Fourth-order

Figure 1
Numerical Solutions to the Euler Equations for the Riemann Problem on a Grid of 50 Mesh Points

The oscillations near the shock are caused by errors in the high frequencies where all the methods do poorly. To efficiently resolve the gradients in this problem, it would be best to adaptively refine the mesh around the discontinuities [5]. These methods either continuously redistribute a fixed number of mesh points to approximate the solution in some near optimal way as the solution changes in time [6] or they add and remove mesh points as needed to minimize the work required to approximate the solution within some prescribed accuracy [7]. This is done by equidistributing a mesh function that reflects the errors in a calculation on a given grid.

BOUNDARY CONDITIONS

Before calculating the solution to any differential equation, boundary conditions should be consistent to form a well-posed problem. A numerical method cannot generate reasonable results for a problem that does not have a well-defined reasonable solution. The importance of proper boundary conditions cannot be overstressed: boundary conditions exert one of the strongest influences on the behavior of the solution. Also, the errors introduced into the calculation from improper boundary conditions persist even as the mesh spacing tends to zero.

A common error in prescribing boundary conditions for hyperbolic equations is to over- or underspecify the number of boundary conditions. Overspecification usually causes nonsmooth solutions with mesh oscillations near the boundary. Underspecification does not ensure a unique solution, and the numerical solution may tend to wander in steady-state calculations. In either case, the results are not accurate and one should be skeptical of even the qualitative behavior of the solution.

It should be noted that the way in which boundary conditions are specified for the difference equations can change a well-posed continuous problem into an ill-posed (unstable) discrete problem. However, our experience with the suggested implementations of following technique has been favorable.

Two of the most common methods used to incorporate boundary conditions into discrete equations are the extrapolation and uncentered difference methods [4].

Extrapolation Method

In this method, the domain of the problem is extended and the solution is extrapolated to fictitious points outside the integration region. The nonphysical solution at these points is defined so that the discrete equations are consistent with as many relationships as can be derived from the boundary conditions and differential equations. The extrapolation formula can do this best by incorporating the discrete boundary conditions [Eq. (3)] into the extrapolant. Additional relations can be generated by differentiating Eq. (2) with respect to time, replacing all time derivatives by space derivatives using Eq. (1), and discretizing the resulting equations.

For example, use a centered five-point formula to approximate the spatial derivative in the equation

$$U_t = (U^2)_{xx} \ , \quad x \in [0,1] \ , \tag{15}$$

with the boundary conditions

$$B(u(x,t),t) = 0, \ x = 0,1 \ . \tag{16}$$

The five-point formula at $x_1 = \Delta x$ uses the nonexistent mesh point $x_{-1} = -\Delta x$. The value $U(x_{-1})$ needed here, is constructed explicitly with an extrapolation formula as follows; Eq. (16) is differentiated with respect to time to give

$$\frac{\partial B}{\partial u}(U(0,t),t)U_t + \frac{\partial B}{\partial t}(U(0,t),t) = 0$$

or

$$\frac{\partial B}{\partial u}(U^2)_{xx} + \frac{\partial B}{\partial t} = 0 \ . \tag{17}$$

Discretizing Eq. (17) with Eq. (6) and rearranging yields the extrapolation formula

$$U_{-1} = \text{known data} + O(\Delta x^4) \ . \tag{18}$$

Uncentered Differences

The second approach is to extend the number of boundary conditions so that all components of the solution are defined at the boundary. Again, these additional boundary conditions must be consistent with the original problem and as many relationships as can be derived from it. An uncentered difference approximation is then used to approximate the spatial derivatives at the mesh points nearest the boundary.

This method is described for the linear hyperbolic system of M equations

$$u_t = h(x)u_x \ , \tag{19}$$

with the boundary conditions

$$su_o = b(t), \ x = x_o \ . \tag{20}$$

Difficulties arise in defining the solution at the boundary when $0 < \text{Rank}(s) < \text{Rank}(h) = M$, and there is no unique solution u_o of Eq. (20). If $\text{Rank}(s) = 0$, all the characteristics are outgoing, and using either uncentered differences at the points near the boundary or straightforward polynomial extrapolation to the fictitious points gives accurate results. When Rank(s) = M, all the characteristics are entering the boundary and all the solution components can be solved for on the boundary. Uncentered spatial differences can then be used at the points near the boundary and will result in an accurate approximation of the boundary conditions. When Rank(s) is greater than zero but less than M, then by differentiating Eq. (20) with respect to time and replacing u_t from Eq. (19), we have

$$sh(x)u_x = b'(t), \quad x = x_o \quad . \tag{21}$$

Approximating U_x by second-order one-sided differences gives us

$$SH_oU_o = [SH_o(4U_1 - U_2) - 2\Delta x b'(t)]/3 + O(\Delta x^3) \quad , \tag{22}$$

where $H_o = H(x_o)$. Equation (22) gives additional information about the boundary conditions that is consistent with both the original boundary conditions of Eq. (20) and the differential Eq. (19). If we still do not have enough boundary conditions to solve for U_o uniquely, we can continue by differentiating Eq. (21) with respect to time and using Eq. (19) again.

Often, H_o is nonlinear and the above procedure must be iterated. Usually one or two iterations will supply a stable accurate boundary approximation.

Once U_o has been found, we can use uncentered finite differences to approximate the spatial derivatives at the mesh points near the boundary or we can extrapolate the solution to fictitious points outside the integration region by replacing the derivatives in Eq. (21) with second-order centered differences and solving for U_{-1}.

TIME DISCRETIZATION

The numerical solution of Eq. (2) is advanced in time in discrete steps that vary depending on the local behavior of the solution; that is, the length of the time steps depends on whether the solution is evolving on a slow or fast time scale. The methods that approximate the time derivatives, like those that approximate the space derivatives, are based on Taylor series. The major difference between time and space differentiation is that time has a direction. This time flow allows savings in computer storage, but introduces questions about the time stability of the difference equations relative to the stability of the differential equation.

The integration methods discussed in this section are called k-cycle Runge-Kutta multistep methods and can be written

$$U^{n+a_1} = \sum_{i=0}^{k_1} \alpha_{1,i} U^{n-i} + \Delta t \sum_{i=0}^{k_2} \beta_{1,i} F^{n-i} + \Delta t \sum_{i=1}^{k} \gamma_{1,i} F^{n+a_i}$$

$$\vdots \qquad\qquad (23)$$

$$U^{n+a_k} = \sum_{i=0}^{k_1} \alpha_{k,i} U^{n-i} + \Delta t \sum_{i=0}^{k_2} \beta_{k,i} F^{n-i} + \Delta t \sum_{i=1}^{k} \gamma_{k,i} F^{n+a_i}$$

$$U^{n+1} = U^{n+a_k}$$

Here, U^{n+a} and $F^{n+a} = F(U^{n+a})$ are approximations to the solution at time $t_{n+a} = t_n + a(t_{n+1} - t_n) = t_n + a\Delta t$, where a is a scalar.

The method is <u>explicit</u> if $\gamma_{i,j} = 0$ for $j \geq i$ and <u>implicit</u> otherwise. Implicit methods require solving one or more algebraic systems on each time step. The extra work required to solve these systems is often rewarded by a substantially larger stability region than a similar explicit method might have.

Explicit Methods

The simplest integration method, called the forward Euler method

$$U^{n+1} = U^n + \Delta t\, F(U^n) \qquad (24)$$

is linearly stable if Δt is chosen so that $\lambda\Delta t$ lies within the stability region shown in Fig. 2. Here, λ is any of the eigenvalues of the linearized Jacobian matrix of F.

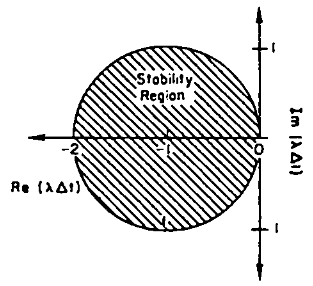

Figure 2
Stability Region for the Forward Euler Method Eq. (24)

If F in Eq. (25) is a second-order finite difference approximation [Eq. (6)] to the parabolic equation

$$u_t = su_{xx} \tag{25}$$

with periodic boundary conditions and an equally spaced mesh between $[-\pi,\pi]$,

$$\lambda = -\frac{2s}{\Delta x^2}(1 - \cos\Theta) \quad , \tag{26}$$

where Θ takes on discrete values between 0 and 2π [3]. From Eq. (26) and Fig. 2, it follows that the stability restriction for Δt is

$$|\Delta t \, \lambda_{max}| \leq 2 \quad , \tag{27}$$

or

$$\frac{\Delta t}{\Delta x^2} s \leq \frac{1}{2} \quad . \tag{28}$$

This restriction is called the Courant-Friedricks-Lewy or CFL condition. We can vary Δt to retain a stable numerical solution as long as Eq. (28) is not violated.

Equation (24) is a first-order integration method. As in the spatial discretization section, higher order methods are often more cost-effective. If Eq. (24) is used to predict U^{n+1}, then this value can be corrected to second order using the improved Euler corrector method.

$$U^{n+1} = U^n + \frac{\Delta t}{2} (\tilde{F}^{n+1} + F^n) \quad . \tag{29}$$

An excellent explicit method for hyperbolic equations is the second-order leap-frog predictor method

$$\tilde{U}^{n+1} = U^{n-1} + 2\Delta t F^n \quad , \tag{30}$$

combined with the third-order leap-frog correction method [4]

$$U^{n+1} = \frac{4}{5}U^n + \frac{1}{5}U^{n-1} + \frac{2\Delta t}{5} \tilde{F}^{n+1} + \frac{4\Delta t}{5} F^n \quad . \tag{31}$$

The stability region shown in Fig. 3 is very good along the imaginary axis. Note that if one stops after the predictor, the method is unstable when the λ have nonzero real parts, making it unsuitable for parabolic equations.

The stability for a centered finite difference approximation to

$$u_t = vu_x \tag{32}$$

with periodic boundary conditions is determined by [3]

$$\lambda_2 = i \sin\Theta/\Delta x \tag{33}$$

or

$$\lambda_4 = i(8\sin\Theta - \sin 2\Theta)/6\Delta x \quad , \tag{34}$$

where λ_2 and λ_4 are the eigenvalues associated with the second- and fourth-order finite difference approximation Eqs. (4) and (5).

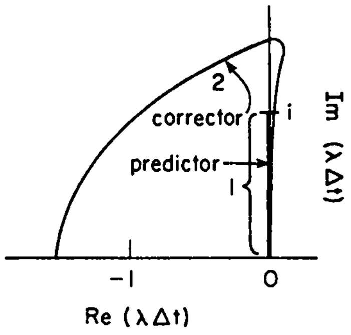

Figure 3
Stability Regions for Leap-frog Predictor Corrector Method

The resulting stability restrictions for the second-order space difference are

$$\frac{\Delta t}{\Delta x} v \leq 1 \qquad \text{predictor stability [Eq. (30)]} \tag{35}$$

$$\frac{\Delta t}{\Delta x} v \leq \frac{3}{2} \qquad \text{predictor-corrector stability [Eqs. (30) and (31)]} \quad . \tag{36}$$

For the fourth-order space difference [Eq. (5)] they are

$$\frac{\Delta t}{\Delta x} v \leq \frac{3}{4} \qquad \text{predictor stability [Eq. (30)]} \tag{37}$$

$$\frac{\Delta t}{\Delta x} v \leq \frac{9}{8} \qquad \text{predictor-corrector stability [Eqs. (30) and (31)]} \quad . \tag{38}$$

The leap-frog predictor is unstable for systems of equations with eigenvalues having a nonzero real part. Therefore, when artificial dissipation is added in shock calculations [4], or when the boundary conditions shift the spectrum of the discretized equation, the leap-frog method cannot be used without the corrector cycle. The first corrector application extends the limit on the maximum time step by 50% and improves the method to third order.

Another difficulty of the leap-frog predictor is a unique type of error caused by time and space mesh decoupling. The odd and even points of a mesh are only weakly coupled for even spatial derivatives; errors with frequency 2Δx or 2Δt can degrade the accuracy of the solution with high-frequency noise. The corrector cycle couples the mesh points among the three time levels and prevents this instability.

A relatively new explicit method, called the iterated multistep (IMS) method [8], can be written as

$$U^{n+a_i} = U^{n+a_{i-1}} + \Delta t c_i [F^{n+a_{i-1}} - F^{n+a_{i-2}}] \ . \tag{39}$$

Here, $a_i = 1$ for $i \geq 3$.

An explicit predictor-corrector is used to start the iteration and the c_i are chosen so that Eq. (39) increases the order of the method on each iteration for linear autonomous systems. For example, if Eqs. (24) and (29) start the iteration, then $c_i = 1/i$, $i = 3,4,\ldots.$ If the leap-frog predictor-corrector Eqs. (30) and (31) start the iteration, then $c_3 = 3/10$, $c_4 = 7/30$, $c_5 = 4/21,\ldots.$

In general, the stability of the IMS methods increases on every iteration, as can be seen in Fig. 4. Another advantage is that IMS methods allow for local improvements in the stability and accuracy of the calculation. Only a single time level is used in the iteration, so only those ODE components that have failed to pass some accuracy test need be iterated. That is, by iterating locally in regions of rapid changes such as shock fronts, boundary layers, or regions with a refined mesh, the stability and accuracy of the calculation are improved precisely where needed.

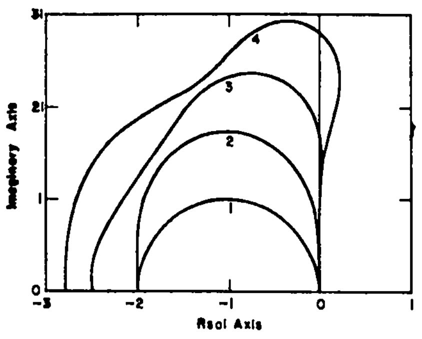

Figure 4a

Stability region for the Improved Euler IMS

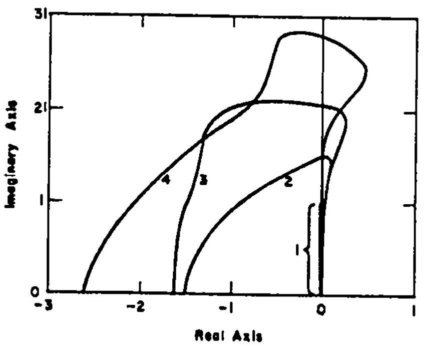

Figure 4b

Stability region for the Leap-Frog IMS

Implicit Methods

Many problems occur when the solution changes on a slow time scale but the stability criteria limit the time step far below that needed to retain accuracy. In these cases, it is often best to use a more stable implicit method.

Two of the best implicit methods are the second-order trapezoidal rule

$$U^{n+1} - \frac{\Delta t}{2} F^{n+1} = U^n + \frac{\Delta t}{2} F^n \tag{40}$$

and the second-order backward difference (BDF) formula

$$U^{n+1} - \frac{2}{3}\Delta t\ F^{n+1} = \frac{4}{3}U^n - \frac{1}{3}U^{n-1} \quad . \tag{41}$$

These methods are stable when $\mathrm{Re}(\lambda) < 0$ for all Δt, as can be seen in Fig. 5.

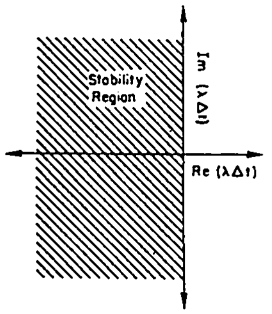

Figure 5a
Trapezoidal Method

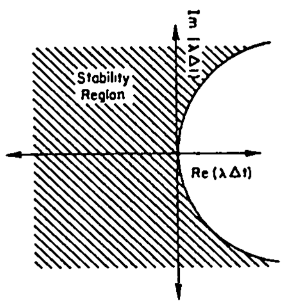

Figure 5b
Second-order BDF

Figure 5
Stability Regions for Trapezoidal Method Eq. (40)
and Second-Order BDF Eq. (41) Method

On each time step of a one-cycle implicit method, we must solve a nonlinear algebraic system of the form

$$U^{n+1} + \Delta t \gamma\ F(U^{n+1}) = \text{known quantities} \quad . \tag{42}$$

Several iterative methods, discussed in the next section, show how Eq. (42) might be solved. A good first guess can often be made by using an explicit or extrapolation method.

ALGEBRAIC SYSTEMS

Iterative Methods

In the discussions on space and time discretization, it became necessary to solve large sparse algebraic systems of equations, which can be written

$$A(v) - b = 0 \quad , \tag{43}$$

where A is a nonlinear discrete operator, b is a known vector, and the discrete solution vector is v.

Often the solution of Eq. (43) is difficult to obtain directly, but the residual error

$$r = A(w) - b \tag{44}$$

for an approximate solution w is easy to evaluate. If there is a related system

$$P(w) - b = 0 \tag{45}$$

that approximates Eq. (43) and is easier to solve, the defect correction algorithm may be appropriate.

Given a guess v_n near a root v_{n+1} of Eq. (43), we can expand Eq. (43) using Taylor series to get

$$\begin{aligned} 0 &= A(v_{n+1}) - b \\ &= A(v_{n+1}) - b + P(v_{n+1}) - P(v_{n+1}) \\ &= A(v_n) - b + P(v_{n+1}) - P(v_n) - (J_P - J_A)(v_{n+1} - v_n) + O(\varepsilon^2) \quad , \end{aligned} \tag{46}$$

where $\varepsilon = v_{n+1} - v_n$. The defect correction iteration is any $O(\varepsilon)$ approximation to Eq. (46). The simplest such iteration is

$$P(v_{n+1}) = P(v_n) - A(v_n) + b \quad . \tag{47}$$

This iteration will converge if v_n and J_P the Jacobian of P, are near enough to v_{n+1} and J_A, respectively. Table 2 lists some of the more common applications of defect corrections.

The iteration (47) can often be speeded up by using a one-step acceleration parameter ω to give

$$P(v_{n+1}) = P(v_n) - \omega_n[A(v_n) - b] \quad . \tag{48}$$

These methods include successive overrelaxation, dynamic alternating direction implicit methods, and damped Newton. Often a two-step acceleration method

$$P(v_{n+1}) = b + \omega_n[P(v_n) - \alpha_n A(v_n)] + (1 - \omega_n)P(v_{n-1}) \tag{49}$$

can speed up the convergence even more. These methods include the Tchebyshev [12] and conjugate-gradient [11] methods.

$P(v_{n+1}) =$	Method
$A(v_n) + J_A(v_n)(v_{n+1} - v_n)$	Newton [9]
Diagonal of J_A	Jacobi [9]
Lower triangular part of J_A	Gauss-Siedel [9]
Lower triangular part of J_A + first upper off-diagonal	Line Gauss-Siedel [9]
Coarse grid operator + relax using one of the above	Multigrid [10]
Symmetric part of J_A	Concus-Golub-O'Leary [11]
If $A = (I + \Delta t\, L_x + \Delta t\, L_y)$, then $\underset{\sim}{P} = (I + \Delta t\, \underset{\sim}{L}_x)(I + \Delta t\, \underset{\sim}{L}_y)$, where $\underset{\sim}{L}_x$ = linearized lower order approximation to L.	ADI [9]
LU where L = lower triangular matrix U = upper triangular matrix	Incomplete LU method [9]

Table 2. Common examples of the defect correction iteration.

NONLINEAR TROUBLES

When using linear methods to approximate the solution of well-posed linear equations, one can have some confidence in the results, thanks to Lax's equivalence theorem. This theorem states that, for these problems, a stable consistent approximation is necessary and sufficient for the numerical approximation to converge to the true solution as the mesh is refined.

The theorem is false for general nonlinear equations or nonlinear methods, as shown in the following two examples.

Nonlinear Equation Example

Consider the nonlinear diffusion equation

$$u_t = (u^3)_{xx}, \quad u(x,0) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \quad . \tag{50}$$

The solution to this equation is a wave front traveling to the right.

Equation (50) can be rewritten as

$$u_t = 3u(u\, u_{xx} + 2u_x^2) \quad . \tag{51}$$

The solution U is discretized and the space derivatives are approximated by any method in the Space Discretization section. The time variable is integrated with any stable method from the Time Discretization section.

Note that $U_t(x_i) = 0$, a discrete version of Eq. (51), is zero for all t when $x_i > 0$ since $U_i \doteq 0$ at these mesh points. This implies that the wave can never propagate to the right of zero, no matter how fine a mesh is used.

Thus, we have a stable consistent approximation that can never converge to the true solution. If Eq. (50) is differenced directly using Eq. (6) with $G = U^3$ then the resulting numerical solution will converge to the correct answer.

Nonlinear Method Example

Consider the unidirectional wave equation

$$u_t = u_x \ , \ u(x,0) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{52}$$

with the solution $u(x,t) = u(x+t,0)$.

After discretizing, we approximate the space derivative using a nonlinear transformation method (Space Discretization, Grid Resolution). The transform interpolates the discrete solution with a monotonicity-preserving interpolant that has a continuous first derivative [13]. The derivative of this interpolant is then used at the grid points to define the ODEs [Eq. (3)].

We know that the true solution to Eq. (52) is monotone and we might expect that an interpolant that preserves this monotonicity will greatly improve the accuracy of the calculation. However, since the interpolant is monotone, $U_x(x_i) = 0$, we have $U_t(x_i) = 0$ at all the grid points. The solution never changes!

SUMMARY

We have used a modular approach to develop accurate and robust methods for the numerical solution of PDEs. The methods to discretize the spatial operator, the boundary conditions, and the time variable, and solve any algebraic system that may arise are combined when writing a code to solve the PDE system. As the last two examples show, special care must be taken when solving a nonlinear equation or when using a nonlinear method. This means that the code must be field tested.

The field test is to check the reliability of the method on a particular nonlinear system of PDEs. The numerical results should be insensitive to reformulations of the equations, small changes in the initial conditions, the mesh orientation and refinement, and the choice of a stable accurate discretization method.

Another excellent analysis tool is verification that any auxiliary relationships (such as conservation laws) hold for the numerically generated solution. These checks should be made -- even if one is absolutely, positively sure that the numerical solution and coding are correct.

When the above checks are made, the methods given here can be combined to give reliable efficient methods for solving a fairly large class of nonlinear PDEs. Also, by using the modular approach presented here, these codes can evolve efficiently since new methods can be quickly incorporated into the program.

ACKNOWLEDGEMENT

I am grateful to Joe Dendy, Blair Swartz, and Burton Wendroff for providing much welcome advice in our many discussions during this work.

REFERENCES

[1] Wachspress, E. L., Iterative solution of elliptic systems (Prentice-Hall, Inc., Englewood Cliffs, NJ, 1966).

[2] Ciment, M., Leventhal, S., and Weinberg, B., "The operator compact implicit method for parabolic equations," J. Comp. Phys. 28 (1978) 135-166.

[3] Hyman, J. M., The method of lines solution of partial differential equations, Courant Institute of Mathematical Sciences report COO-3077-139 (1976).

[4] Hyman, J. M., A method of lines approach to the numerical solution of conservation laws, Adv. in Comp. Methods for PDEs - III, Vichnevetsky, R., and Stepleman, R. S., (eds) Publ. IMACS (1979) 313-321.

[5] Hyman, J. M., "The numerical solution of time dependent PDEs on an adaptive mesh," Los Alamos Scientific Laboratory LA-UR-80-3702 (1980).

[6] Gelinas, R. J., Doss, S. K., and Miller, K., "The moving finite element method: applications to general partial differential equations with multiple large gradients," J. Comp. Phys. 40 (1981) 202-249.

[7] Berger, M., Gropp, W., and Oliger, J., Grid generation for time dependent problems: criteria and methods, numerical grid generation techniques, NASA Conf. Proc. Publication 2166 (1980) 181-188.

[8] Hyman, J. M., Explicit A-stable methods for the solution of differential equations, Los Alamos Scientific Laboratory LA-UR-79-29 (1979).

[9] Young, D. M., Iterative solution of large linear systems (Academic Press, New York, NY, 1971).

[10] Brandt, A., Multi-level adaptive solutions to boundary value problems, Math. Comp. 31 (1977) 333-390.

[11] Concus, P., Golub, G. H., and O'Leary, D. P., Numerical solution of nonlinear elliptic partial differential equations by the generalized conjugate gradient method, Computing 19 (1978) 321-339.

[12] Manteuffel, T. A., The Tchebychev iteration for nonsymmetric linear systems, Numer. Math. 28 (1977) 307-327.

[13] Hyman, J. M., Accurate monotonicity preserving cubic interpolation, Los Alamos National Laboratory report LA-8796-MS (1981).

Page Range	Domestic Price	NTIS Price Code	Page Range	Domestic Price	NTIS Price Code	Page Range	Domestic Price	NTIS Price Code	Page Range	Domestic Price	NTIS Price Code
001-025	$ 5.00	A02	151-175	$11.00	A08	301-325	$17.00	A14	451-475	$23.00	A20
026-050	6.00	A03	176-200	12.00	A09	326-350	18.00	A15	476-500	24.00	A21
051-075	7.00	A04	201-225	13.00	A10	351-375	19.00	A16	501-525	25.00	A22
076-100	8.00	A05	226-250	14.00	A11	376-400	20.00	A17	526-550	26.00	A23
101-125	9.00	A06	251-275	15.00	A12	401-425	21.00	A18	551-575	27.00	A24
126-150	10.00	A07	276-300	16.00	A13	426-450	22.00	A19	576-600	28.00	A25
									601-up	†	A99

†Add $1.00 for each additional 25-page increment or portion thereof from 601 pages up.