# ALGORITHMS AND SOFTWARE FOR ORDINARY DIFFERENTIAL EQUATIONS AND DIFFERENTIAL-ALGEBRAIC EQUATIONS, PART I: EULER METHODS AND ERROR ESTIMATION

**Alan C. Hindmarsh and Linda R. Petzold**

**Department Editor: William J. Thompson**

*This two-part article describes methods for solving systems of differential equations. Part I reviews the explicit Euler method, discusses "stiffness," and describes how and why the implicit Euler method can provide useful solutions of stiff systems. Part I concludes with a consideration of errors and error estimates. Part II, which will appear in the next issue, extends the discussion to higher-order methods of both the multistep and one-step varieties. Part II gives special attention to large stiff systems and differential-algebraic systems. The article concludes with a description of relevant software packages that are freely available from Netlib on the Internet.*

Gaining insight into a physical process is frequently accomplished by constructing a mathematical model and computing solutions to it. Very often such a model takes the form of a system of differential equations that govern the behavior of the relevant physical variables as a function of time. If these variables are also functions of space, then in the computation the continuous spatial coordinates must also be discretized in some way. Problems of this sort arise in a wide variety of disciplines. Among the scientific areas that generate time-dependent differential-equation problems are chemical kinetics, laser kinetics, mechanical systems, molecular dynamics, power systems, neuronal modeling, electronic networks, computational fluid dynamics, and various reaction-transport processes.

It is rare that a realistic mathematical model is amenable to a purely analytic solution. We must usually generate a computational model from the mathematical one. While avoiding issues of analytic solvability, this introduces a variety of other difficult issues that couple the features of the original model and the computing environment. On recognizing that problems arising in various disciplines share many formal mathematical properties, the field of numerical mathematics seeks to devise powerful and general techniques for the transformation of a mathematical to a computational model, and for the efficient numerical solution of the latter. As a result, many of the differential equation problems that arise in applications are now routinely solved by the use of general-purpose mathematical software packages. The availability of such software has the additional advantage of leaving the scientist free to focus on the content of the model itself instead of the details of its numerical solution.

The effort represented by modern numerical algorithms and software goes far beyond what could be justified within any one discipline or application that benefits from it. A typical ordinary-differential-equation (ODE) solver available today might well represent several man-years of work just in the development and testing of the computer code, excluding many previous man-years of theoretical investigations into error estimation, numerical stability, and efficiency. The effort leading to a production code is highly cost-effective because it benefits a broad spectrum of users.

In what follows, we identify some of the more important issues in solving problems involving differential equations, show how these ideas lead to various kinds of solution methods, and outline the current state of research work in these areas.

*Alan C. Hindmarsh is a mathematician in the Center for Computational Sciences and Engineering at Lawrence Livermore National Laboratory, Livermore, CA 94550. Linda R. Petzold is a professor in the Department of Computer Science, University of Minnesota, Minneapolis, MN 55455. Hindmarsh and Petzold have been responsible for developing numerous software packages for the solution of ordinary differential equations and differential-algebraic equations.*

## Systems of differential equations

Mathematical models frequently take the form of a system of ODEs, and for the moment we will suppose that these can be written in the concise and explicit general form

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}). \tag{1}$$

Here, $t$ is time and $\mathbf{y}$ is a vector of dependent variables of interest (the "state variables"). To save writing, we will often denote $d\mathbf{y}/dt$ by $\mathbf{y}'$. The *initial value problem* for Eq. (1) is to find the solution $\mathbf{y}(t)$ that satisfies a given *initial condition* $\mathbf{y}(t_0)=\mathbf{y}_0$.

In many instances, the model also involves state variables whose time derivatives do not appear in the equations. Then the set of equations is known as a *differential-algebraic equation* (DAE) system. The most general DAE system is written as

$$\mathbf{F}(t,\mathbf{y},\mathbf{y}')=0, \tag{2}$$

where $\mathbf{F}$ is some function. An important special case is the "semiexplicit" system

$$\begin{aligned}\frac{d\mathbf{y}}{dt}&=\mathbf{f}(t,\mathbf{y},\mathbf{z}),\\ 0&=\mathbf{g}(t,\mathbf{y},\mathbf{z}),\end{aligned} \tag{3}$$

where $\mathbf{z}$ is another vector of dependent variables. Here, $\mathbf{z}$ is coupled to the ODE for $\mathbf{y}$, but $d\mathbf{z}/dt$ does not appear.

The independent variable $t$ need not actually be time, of course. ODE and DAE initial value problems arise in applications in which the independent variable is a spatial coordinate or some other variable, and everything we say applies equally well to such problems. Yet in practice the majority of these problems actually do involve time, and the nomenclature in the literature on numerical ODE and DAE methods reflects that fact, in terms such as "time step" and "time integration."

## Example: An ozone model

In order to give an idea of the kinds of problems for which ODE and DAE solvers can be effectively applied, we give here an example problem derived from a time-dependent system of PDEs. The problem comes from atmospheric modeling, namely the production and transport of ozone in the stratosphere. However, it has been considerably simplified in order to make it presentable in full in a limited space such as this.

The model is a system of two coupled PDEs in time and two space dimensions. The dependent variables represent, respectively, the concentrations ($c^i$, with $i=1,2$) of the species $O_1$ (singlet oxygen) and $O_3$ (ozone) in mol/cm$^3$. Molecular oxygen $O_2$ is of course also present, but is assumed to have a constant concentration of $3.7\times10^{16}$ here. The kinetic interaction between the three species is governed by the so-called Chapman mechanism, which includes the destruction of ozone by sunlight:

$$\begin{aligned}&O+O_2\rightarrow O_3, \quad k_1,\\ &O+O_3\rightarrow 2O_2, \quad k_2,\\ &O_2\rightarrow 2O, \quad k_3(t),\\ &O_3\rightarrow O+O_2, \quad k_4(t).\end{aligned}$$

Thus the chemistry is diurnal, having a reaction rate constant that varies with the time of day. In addition, vertical and horizontal diffusion is assumed, with the vertical diffusivity increasing with altitude. Specifically, the PDE system is

$$\frac{\partial c^i}{\partial t}=K_h\frac{\partial^2 c^i}{\partial x^2}+\frac{\partial}{\partial z}\left(K_v(z)\frac{\partial c^i}{\partial z}\right)+R^i(c^1,c^2,t)$$

$$(i=1,2),$$

$$K_h=4\times10^{-6}\ \text{km}^2/\text{s} \quad \text{and} \quad K_v(z)=10^{-8}e^{z/5}\ \text{km}^2/\text{s}.$$

The spatial domain is the rectangle $0\leq x\leq 20$, $30\leq z\leq 50$ km ($x$=latitude, $z$=altitude), and the time interval is $0\leq t\leq 432\,000$ s (5 days). The reaction terms are given by:

$$R^1(c^1,c^2,t)=-k_1c^1-k_2c^1c^2+k_3(t)7.4\times10^{16}+k_4(t)c^2,$$

$$R^2(c^1,c^2,t)=k_1c^1-k_2c^1c^2-k_4(t)c^2,$$

$$k_1=6.031, \quad k_2=4.66\times10^{-16},$$

$$k_3(t)=\begin{cases}\exp[-22.62/\sin(\omega t)], & \text{for } \sin(\omega t)>0,\\ 0, & \text{for } \sin(\omega t)\leq 0,\end{cases}$$

$$k_4(t)=\begin{cases}\exp[-7.601/\sin(\omega t)], & \text{for } \sin(\omega t)>0,\\ 0, & \text{for } \sin(\omega t)\leq 0,\end{cases}$$

$$\omega=\pi/43{,}200\ \text{s}^{-1}.$$

As boundary conditions, we pose homogeneous Neumann boundary conditions (zero gradients) on all boundaries. To complete the problem description, we pose initial profiles for $c^i$ at $t=0$ which are consistent with the boundary conditions.

The process of generating an ODE system for this PDE problem is referred to as semidiscretization, and also as the method of lines. We discretize the spatial region, with (in this case) a uniform mesh of size $M\times M$. At each mesh point $(x_j, z_k)$ we have approximate values $c^i_{j,k}$ for the two concentrations. Central differencing gives discrete approximations to the spatial derivatives in the PDEs. The result is an ODE system in the vector

$$\mathbf{y}=(c^1_{1,1},c^2_{1,1},c^1_{2,1},c^2_{2,1},\ldots,c^2_{M,M},c^2_{M,M})^T$$

of length $2M^2$. The ordering is first by species index, then by $j$, then by $k$. Initial conditions for the ODE system would simply be the discrete values of the given initial profiles for the $c^i(x,z)$.

The ODE initial value problem obtained in this example has some interesting features. First, the size of the problem can be arbitrarily large, depending on $M$. Second, the rate constants $k_i$ span a considerable range of values, and as a result the ODE system has the property of "stiffness," which will be discussed in detail shortly. Third, the diurnal variation of the last two rate constants causes a corresponding wide diurnal variation in one of the solution components (oxygen singlet). The diurnal variation of the rate constant $k_3$ over a five-day period is shown in Fig. 1. The combination of these properties makes the problem particularly challenging for an ODE solver.

## The Euler method

To illustrate the variety of issues associated with ODE

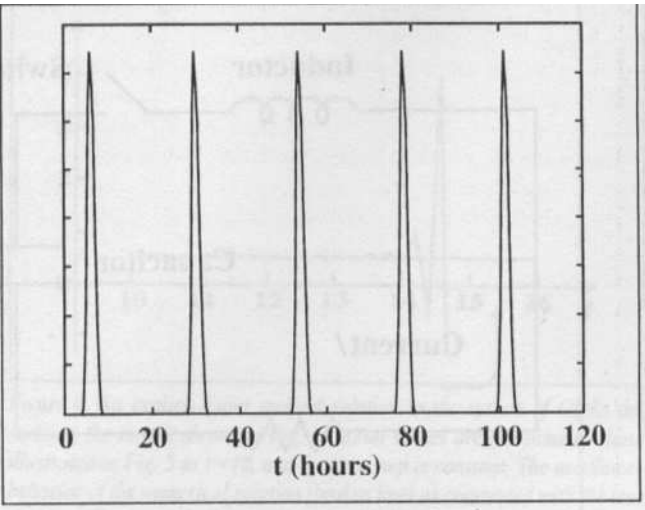


*Figure 1. The diurnal kinetic rate constant $k_3(t)$ is shown over a five-day period. It peaks at noon of each day and is zero during the night time. The variation in time of the concentration of $O_1$ follows the same diurnal pattern very closely.*

and DAE problems, we examine some typical numerical methods. The oldest and simplest of all methods for solving ODEs was devised by the mathematician Leonhard Euler in the 18th century. It consists of computing discrete vectors $\mathbf{y}_1,\mathbf{y}_2,\ldots$, that approximate $\mathbf{y}(t)$ at the times $t_1,t_2,\ldots$, starting from the initial condition $\mathbf{y}(t_0)=\mathbf{y}_0$. If $\mathbf{y}_{n-1}$ has been computed for some $n\geq 1$, then $\mathbf{y}_n$ is defined as

$$\mathbf{y}_n=\mathbf{y}_{n-1}+h_n\mathbf{f}(t_{n-1},\mathbf{y}_{n-1}), \tag{4}$$

where $h_n=t_n-t_{n-1}$ is the size of the time step. In other words, the next solution point is computed at time $t_n=t_{n-1}+h_n$ by moving from the point $(t_{n-1},\mathbf{y}_{n-1})$ on a line at a constant slope of $\mathbf{f}(t_{n-1},\mathbf{y}_{n-1})$, the slope of the solution through that point according to Eq. (1). The method is completely *explicit*: The new value is defined directly in terms of the known previous values. This leaves unspecified the choice of the step sizes $h_1,h_2,\ldots$, but we defer this question until later. Figure 2 shows this Euler-method solution (connected dots) for a single ODE, along with the true solution (solid curve). In this case, the step sizes $h_n$ are all equal.

Although the Euler method is natural and easy to apply, it is rarely the method of choice, for reasons that will become clear later on. As suggested by Fig. 2 the numerical solution can easily drift away from the true solution unless the step sizes are kept quite small. Suppose we use the Euler method to solve Eq. (1) from $t_0$ to a fixed final time $T$ with $N$ steps of equal size $h=(T-t_0)/N$, and that we let $N\to\infty$, so that $h\to 0$. If we suppose also that we know the exact final answer $\mathbf{y}(T)$, then we would find that the error in the final computed value $\mathbf{y}_N$ behaves as

$$\mathbf{y}_N-\mathbf{y}(T)=O(h). \tag{5}$$

In fact, this general behavior of the error can be deduced by a careful analysis. We will see later that, even on problems in which the Euler solution appears to be reasonably accurate, much better error behavior can be achieved with other methods [for example, error$=O(h^2)$] at very little additional cost. For reference, the dominant cost in this Euler solution is $N$ evaluations of $\mathbf{f}$ (one evaluation per step).

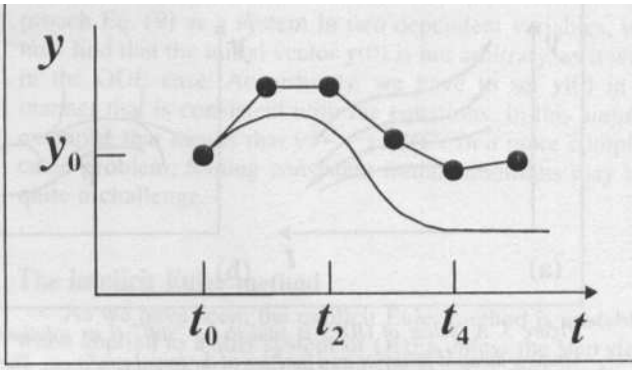


*Figure 2. The Euler method is the oldest and simplest technique for solving ODEs. Shown here are the numerical solution computed by the Euler method (connected dots) and the true solution (solid curve). Since the computed solution can quickly drift away from the true solution unless the step sizes are quite small, it is no longer the method of choice.*

The Euler method is not directly applicable to DAE systems, even in the special semiexplicit case of Eq. (3). If we have values $\mathbf{y}_{n-1}$ and $\mathbf{z}_{n-1}$ approximating $\mathbf{y}$ and $\mathbf{z}$ at time $t=t_{n-1}$, we can apply the Euler method in Eq. (4) to the ODE of Eq. (3) to advance $\mathbf{y}$ to $\mathbf{y}_n$, but there is no easy way to advance $\mathbf{z}$. We might pose the problem of solving the algebraic equation

$$\mathbf{g}(t_n,\mathbf{y}_n,\mathbf{z}_n)=0 \tag{6}$$

for $\mathbf{z}_n$ (given $t_n$ and $\mathbf{y}_n$), but this may be either difficult because of the nonlinear way in which $\mathbf{g}$ depends on $\mathbf{z}$, or even mathematically impossible, because the dependence of $\mathbf{g}$ on $\mathbf{z}$ may be singular (unsolvable). In an extreme case (which occurs in equations describing incompressible hydrodynamics), $\mathbf{g}=g(t,\mathbf{y})$ does not depend on $\mathbf{z}$ at all, and there is no hope of solving Eq. (6), yet the DAE system [Eq. (3)], is well posed (it has a well-defined solution).

## Stiff systems

Another important issue in matching solution methods to ODE problems is stiffness. In the simplest terms, the ODE system of Eq. (1) is said to be stiff if it has a strongly damped, or "superstable" mode. To get a feeling for the concept, consider the solutions $\mathbf{y}(t)$ of an ODE system starting from various initial conditions. For a typical non-stiff system, if we plot a given component of the vector versus $t$, we might get a family of curves such as those shown in Fig. 3(a). The curves show a stable tendency to merge as $t$ increases, but not very rapidly. When such a family of curves is plotted for a typical stiff system, the result might be as shown in Fig. 3(b). Here, the curves merge rapidly to a set of smoother curves, the deviation from the smooth curve being strongly damped as $t$ increases.

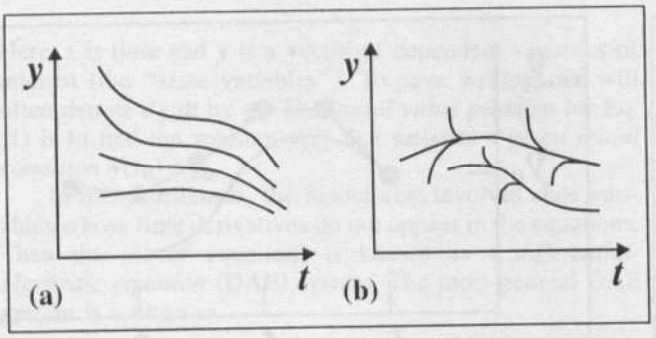


*Figure 3. A system of ODEs is said to be "stiff" if its solutions show strongly damped behavior as a function of the initial conditions. The family of curves shown in (a) represents the behavior of solutions to a nonstiff system for various initial conditions. In contrast, solutions to the stiff system shown in (b) tend to merge quickly.*

Stiffness in a system of ODEs corresponds to a strongly stable behavior of the physical system being modeled. At any given time, the system is in a sort of equilibrium (though not necessarily a static one). Accordingly, if some state variable is perturbed slightly, the system responds rapidly to restore itself to equilibrium. Typically, the true solution $\mathbf{y}(t)$ of the corresponding ODE system shows no such rapid variation, except possibly at the very beginning of the time interval. However, the potential for rapid response is present in the ODEs at all times, and becomes real if one poses an initial value problem by perturbing $\mathbf{y}$ at some point out of equilibrium. The system is said to have at least two time scales (or time constants); by a "time scale," we mean the rough value of the spacing of $t$ values needed to resolve a solution curve accurately. There is a long time scale present in the solution of interest, and there is a short time scale given by the damping time (or time constant) of any of the perturbed solutions. The more different these two time scales are, the stiffer the system is; the ratio of the longest to the shortest time constant in a stiff system is called the "stiffness ratio" of the system.

Stiffness is perhaps the best understood by means of a small example. The simple damped oscillator circuit in Fig. 4, with a capacitor, a resistor, and an inductor, has an electric current $I$ that obeys the second-order ODE

$$L\frac{d^2I}{dt^2}+R\frac{dI}{dt}+\frac{I}{C}=0. \tag{7}$$

If we let $\mathbf{y}$ be a vector with two components, $y^1=I$, and $y^2=dI/dt$ [we use superscripts to avoid confusion with the notation in Eq. (4)], then Eq. (7) is equivalent to a system of the same form as Eq. (1), namely

$$\frac{dy^1}{dt}=y^2,\quad \frac{dy^2}{dt}=-(R/L)y^2-y^1/LC. \tag{8}$$

Consider parameter values such that (in suitable dimensionless units) $R/L=20$ and $LC=100$, and initial conditions at time $t=0$ in which $I=0$ and $dI/dt=10$ (as if a voltage were applied to the circuit and then switched off). In the notation of Eqs. (1) and (4), $t_0=0$ and $\mathbf{y}_0=\binom{0}{10}$.

Figure 5 is a plot of the solution (solid line), where the

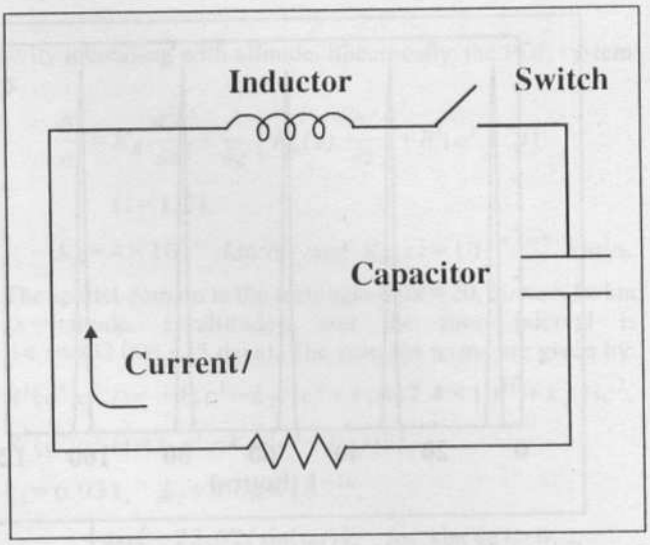


*Figure 4. A simple electrical circuit illustrates the behavior of a typical stiff system. In this case, both the resistor and the capacitor damp perturbations to the system caused by a change in current.*

time axis is logarithmic for convenience. Notice that the solution varies on a time scale of less than 0.1 at early times, then becomes smooth and varies on a time scale of around 1000. The system has two different time scales and a stiffness ratio of around 10 000. In fact, a precise analytic solution is easily derived. It consists of a linear combination of simple exponential functions $\exp(-t/\tau_1)$ and $\exp(-t/\tau_2)$, where (very nearly) $\tau_1=0.05$ and $\tau_2=2000$. The short time constant $\tau_1$ is present in the system even when the solution has a much longer time scale, as can be seen by posing an initial value problem with a perturbed initial $\mathbf{y}$ at (say) $t=10$. Such a perturbed solution is shown as the dashed line in Fig. 5.

The smallest time scale in a stiff system manifests itself in another way when we try to carry out a numerical solu-

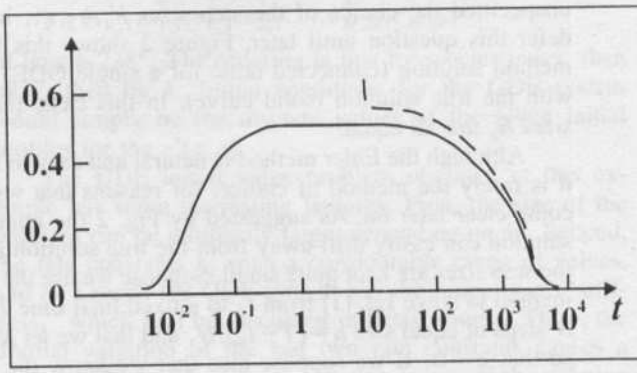


*Figure 5. The solid curve is a plot of the solution to the ODE describin the circuit shown in Fig. 4. Note that the time axis is logarithmic. Th plateau of the curve separates two regions where the solution varies on tw different time scales. If the initial value of the $\mathbf{y}$ variable is perturbed t=10 (dashed curve), the shorter time scale predominates for a while, an then the solution displays the same long time scale as the unperturb solution.*

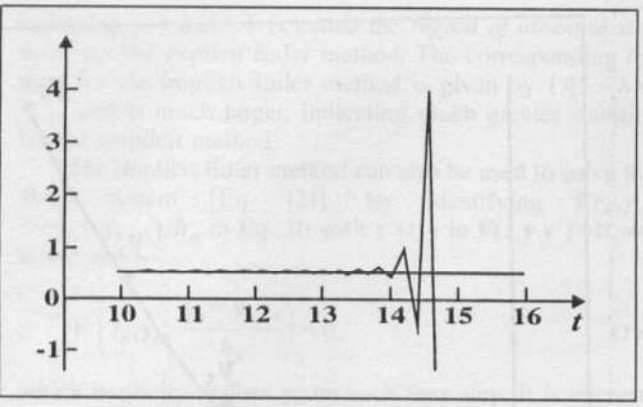


*Figure 6. An explicit Euler method solution to the system of ODEs describing the circuit shown in Fig. 4. Initial values are the same as those illustrated in Fig. 5 at t=10, and the time step is constant. The oscillatory behavior of the numerical solution (broken line) as contrasted with the true solution (flat curve) indicates that the explicit Euler method introduces an instability in this application. An accurate and stable solution by Euler's method would require a step size smaller than the shortest time scale of the problem.*

ion of the system. Solution by an explicit method like the Euler method either will produce completely inaccurate answers or will require very small time step sizes (comparable with the smallest time constant present in the system) to get accurate answers.

Figure 6 shows a partial solution by the Euler method of the problem of Eq. (8), starting with values taken from the earlier one at $t=10$ and using a constant step size $h=0.2$ (broken line), along with the true solution (flat curve). After a while, the successive values of $y^1=I$ oscillate roughly like $(-3)^n$. We say the numerical method is *unstable* when this happens. To get a reasonably accurate and stable Euler method solution of this problem, we must use values of $h$ well below 0.05. Yet this part of the true solution is very well resolved on a time scale of more than 10.

The circuit problem of Eq. (8) also provides an example of a DAE system, albeit a very simple one. If we fix $R$ and $C$ but make the inductance $L$ smaller and smaller, the ODE system Eq. (8) becomes more and more stiff (the stiffness ratio is roughly $R^2C/L$). In the limit $L=0$, Eq. (8) (with the second equation first multiplied by $L$) reduces to the DAE system

$$\frac{dy^1}{dt}=y^2, \quad 0=-Ry^2-y^1/C. \tag{9}$$

Here, no time derivative of $y^2$ appears, and the system has the general form Eq. (3) (with $y=y^1$ and $z=y^2$). The limit process has changed the mathematical properties of the system in a fundamental way: although Eq. (7) or (8) allow us to freely specify two initial conditions ($I$ and $dI/dt$) the system [Eq. (9)] allows only one, since $y^1$ and $y^2$ are algebraically related. This example trivially enables us to eliminate $y^2$, leaving a single first-order ODE, which is the limit of Eq. (7) as $L$ approaches zero. But for a complicated DAE problem, this elimination may be either impossible or highly impractical. So if we continue to approach Eq. (9) as a system in two dependent variables, we now find that the initial vector $\mathbf{y}(0)$ is not arbitrary, as it was in the ODE case. Accordingly, we have to set $\mathbf{y}(0)$ in a manner that is consistent with the equations. In this simple example, that means that $y^2=-y^1/RC$. In a more complicated problem, finding consistent initial conditions may be quite a challenge.

## The implicit Euler method

As we have seen, the explicit Euler method is unstable when applied to a stiff system of ODEs unless the step size is constrained to be smaller than the shortest time scale of the system. This constraint on the step size can be a very severe limitation in some applications, forcing the method to take time steps that are intolerably small before acceptable accuracy is obtained. For some problems, the explicit Euler time steps must be so small (in inverse proportion to the stiffness) that roundoff errors degrade the numerical solution significantly, and the computation cost is prohibitive. It is natural to ask whether there are other methods that can solve stiff systems using time steps that are not limited by stability but only by the need to resolve the solution curve. It is now widely recognized that in general the answer requires the use of implicit methods, and in particular methods that are designed to have good stability properties for stiff systems. The simplest of these methods is the implicit Euler method.

The implicit Euler method for the ODE [Eq. (1)] is given by

$$\mathbf{y}_n=\mathbf{y}_{n-1}+h_n\mathbf{f}(t_n,\mathbf{y}_n). \tag{10}$$

In contrast to the explicit Euler formula (4), this method is called *implicit* because $\mathbf{y}_n$ is not defined directly in terms of past values of the solution. Instead, it is defined implicity as the solution of the nonlinear system of equations [Eq. (10)]. We can write this nonlinear system abstractly as

$$\mathbf{F}(\mathbf{u})=0, \tag{11}$$

where $\mathbf{u}=\mathbf{y}_n$ and $\mathbf{F}(\mathbf{u})=\mathbf{u}-\mathbf{y}_{n-1}-h_n\mathbf{f}(t_n,\mathbf{u})$. The nonlinear system of Eq. (11) is typically solved by Newton iteration,

$$\left(\frac{\partial\mathbf{F}}{\partial\mathbf{u}}\right)(\mathbf{u}^{(m+1)}-\mathbf{u}^{(m)})=-\mathbf{F}(\mathbf{u}^{(m)}). \tag{12}$$

Here, if $N$ is the size of the ODE system, $\mathbf{u}$ and $\mathbf{F}$ are vectors of length $N$, and the Jacobian matrix $\partial\mathbf{F}/\partial\mathbf{u}$ is an $N\times N$ matrix of partial derivatives of $\mathbf{F}$ evaluated at $\mathbf{u}^{(m)}$. Thus, there is a linear system to be solved at each iteration. Newton's method converges in one iteration for linear systems, and the convergence is quite rapid for general nonlinear systems, given a good initial guess. For the initial guess, we can use an explicit formula such as the explicit Euler method or, more commonly, a polynomial that coincides with recent past solution values, evaluated at $t_n$. In practice, the Jacobian matrix is *not* reevaluated at each iteration, and furthermore is often approximated by numerical difference quotients rather than evaluated exactly. This use of an approximate Jacobian that is fixed throughout the

iteration sequence in Eq. (12) is called *modified Newton* iteration.

To gain a better understanding of why the implicit Euler method does not need to restrict the step size to maintain stability for stiff systems, let us consider a very simple example,

$$y' = -\alpha(y - t^2) + 2t, \quad y(0) = 0, \tag{13}$$

on the interval $0 \leq t \leq 1$. Here, $\alpha$ is a positive parameter. When $\alpha$ is very large, the system is stiff. The general solution to Eq. (13) is given by

$$y(t) = t^2 + y_0 e^{-\alpha t}.$$

This equation shows clearly that if $\alpha$ is large and the initial value is perturbed slightly away from $y_0 = 0$, the solution tends rapidly back to the curve $y = t^2$. This behavior is characteristic of stiff systems. A sketch of the solution by the implicit Euler method for a slightly perturbed initial value is given in Fig. 7(a), where it can be seen that the numerical solution exhibits the correct behavior. In contrast, the explicit Euler method solution is shown in Fig. 7(b), where the instability is evident in the same way as in the circuit example (see Fig. 6).

To see why the implicit Euler method gives such a good result for this problem, we can examine the error propagation properties of this method in more detail. When the implicit Euler method is applied to Eq. (13), we obtain

$$y_n = y_{n-1} - h\alpha(y_n - t_n^2) + 2ht_n. \tag{14}$$

(Here we are dropping the subscript on $h$.) If we expand the true solution $y(t)$ in a series about $t_{n-1}$, we find that

$$y(t_n) = y(t_{n-1}) - h\alpha[y(t_n) - t_n^2] + 2ht_n + O(h^2). \tag{15}$$

Subtracting Eq. (15) from Eq. (14) and defining the global error $e_n = y_n - y(t_n)$, we obtain

$$e_n = e_{n-1} - h\alpha e_n + O(h^2). \tag{16}$$

Solving for $e_n$, we see that

$$|e_n| \leq \frac{|e_{n-1}|}{|1 + h\alpha|} + O(h^2). \tag{17}$$

Thus the global error remains small even for large values of $\alpha$. In contrast, the global error for the explicit Euler method satisfies

$$|e_n| \leq |1 - h\alpha| \; |e_{n-1}| + O(h^2). \tag{18}$$

Here the error will grow exponentially unless $|1 - h\alpha| < 1$. Thus the step size must be constrained to satisfy $h \leq 2/\alpha$.

For general ODE systems $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$, the negative of the eigenvalues of the matrix $J = \partial \mathbf{f}/\partial \mathbf{y}$ play the role of $\alpha$. For stiff systems, the eigenvalues of $J = \partial \mathbf{f}/\partial \mathbf{y}$ include at least one with a relatively large negative real part. In the circuit example [Eq. (8)], the eigenvalues of $J$ are approximately $-0.0005$ and $-20.0$. The great disparity between these two numbers is what makes the problem stiff. When $\lambda$ is viewed as an eigenvalue of $J$, the set of complex numbers $h\lambda$

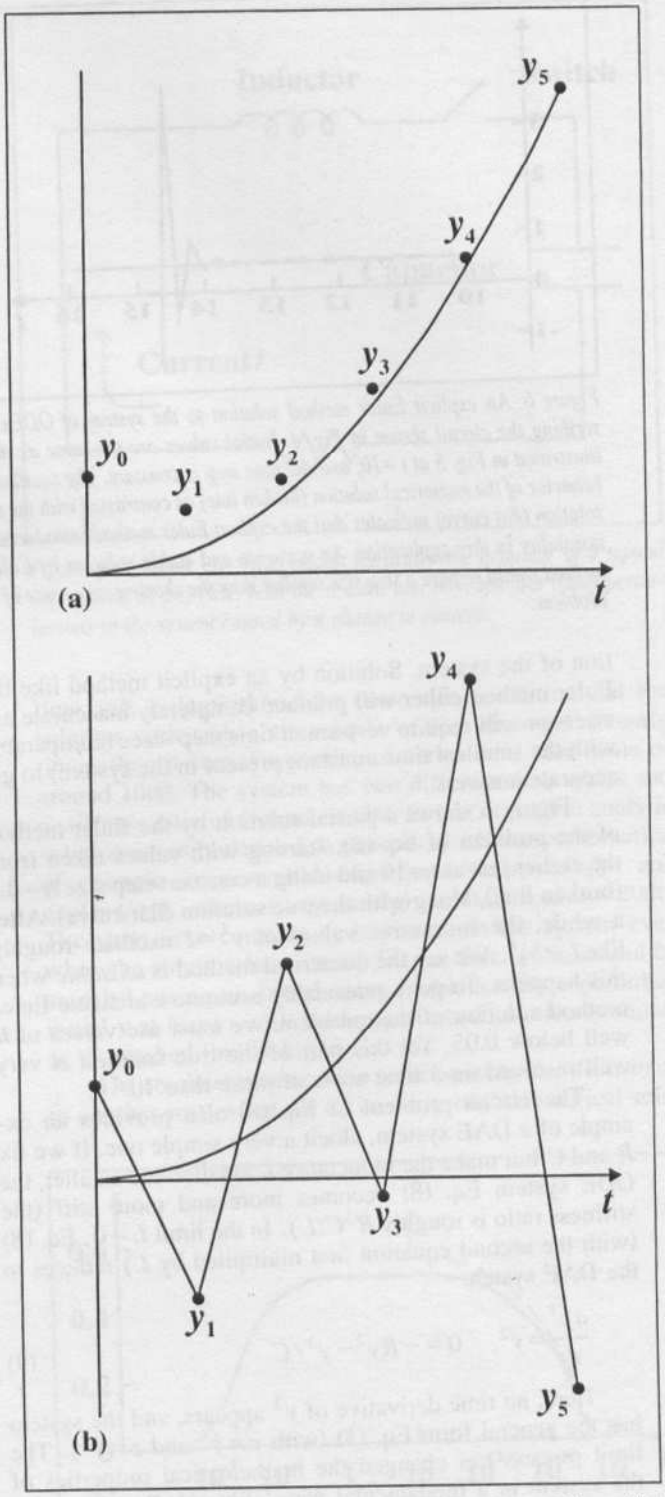


*Figure 7. The implicit Euler method overcomes a weakness of the explicit Euler method in that it does not need to restrict the step size to provide stable solutions for stiff systems. The solution of the system of Eq. (13) for a slightly perturbed initial value, shown in (a), was generated by the implicit Euler method. It is well behaved in the sense that the y values merge rapidly with the unperturbed solution curve. In contrast, the explicit Euler method applied to the same system produces the erratic oscillatory behavior shown in (b).*

satisfying $|1+h\lambda|<1$ is called the *region of absolute stability* for the explicit Euler method. The corresponding region for the implicit Euler method is given by $1/|1-h\lambda|<1$, and is much larger, indicating much greater stability for the implicit method.

The implicit Euler method can also be used to solve the DAE system [Eq. (2)]. By identifying $\mathbf{f}(t_n,\mathbf{y}_n)=(\mathbf{y}_n-\mathbf{y}_{n-1})/h_n$ in Eq. 10 with $\mathbf{y}'(t_n)$ in $\mathbf{F}(t,\mathbf{y},\mathbf{y}')=0$, we arrive at

$$\mathbf{F}\left(t_n,\mathbf{y}_n,\frac{\mathbf{y}_n-\mathbf{y}_{n-1}}{h_n}\right)=0, \tag{19}$$

which implicity defines $\mathbf{y}_n$ on each time step. It is interesting to note that when the implicit Euler method is applied to the very simple DAE system

$$y(t)-t^2=0$$

which is the limit of Eq. (13) as $\alpha\to\infty$, the solution is $y_n=t_n^2$. Thus, the implicit Euler method is exact for this problem! More generally, when applied to the semiexplicit DAE system of Eq. (3), the implicit Euler method yields the pair of equations

$$\frac{\mathbf{y}_n-\mathbf{y}_{n-1}}{h_n}=\mathbf{f}(t_n,\mathbf{y}_n,\mathbf{z}_n),$$

$$0=\mathbf{g}(t_n,\mathbf{y}_n,\mathbf{z}_n),$$

for the new values $\mathbf{y}_n$ and $\mathbf{z}_n$. That is, we replace the ODE by the implicit Euler equation and force the algebraic equation $\mathbf{g}=0$ to hold at the same time. It turns out that the implicit Euler method, as well as some higher-order generalizations of this method, have several properties that make them quite attractive for the solution of DAE systems.

## Errors and error estimates

In the previous section, we derived recurrence relations for the global errors of the implicit and explicit Euler methods applied to a specific stiff ODE. We saw that although the errors remain small for the implicit Euler method, errors for the explicit Euler method can propagate in a disastrous way. It is important in using these methods to have a basic understanding of the various types of errors that are associated with a computation. Modern computer codes attempt to adjust the step size to control the size of some of these errors but not others.

For simplicity, we return to the implicit Euler method applied to the ODE system of Eq. (1)

$$\mathbf{y}_n=\mathbf{y}_{n-1}+h_n\mathbf{f}(t_n,\mathbf{y}_n). \tag{20}$$

On each step, this method makes an error that results from the approximation of the differential equation by the difference equation. One measure of this error is the amount by which the true solution to the ODE fails to satisfy the difference equation defined by the method. This is known as the *local truncation error* or *local discretization error*. For the implicit Euler method, the local truncation error is given by

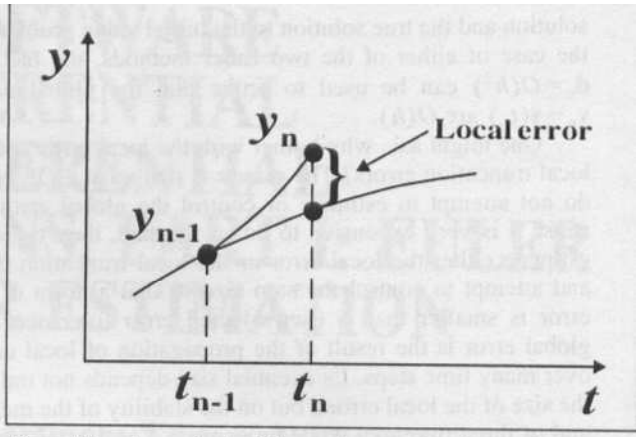


*Figure 8. Local error is the difference between the value $\mathbf{y}_n$ of the numerical solution to an ODE at a time $t_n$ and the value of the true solution that passes through the numerical solution at the last time step $\mathbf{y}_{n-1}$.*

$$\mathbf{d}_n=\mathbf{y}(t_{n-1})+h\mathbf{f}[t_n,\mathbf{y}(t_n)]-\mathbf{y}(t_n),$$

which, after expanding in a series about $t_{n-1}$, we can simplify to

$$\mathbf{d}_n=\frac{h^2}{2}\mathbf{y}''(\xi_n)$$

for some $\xi_n$ in $t_{n-1}<\xi_n<t_n$.

There is another measure of the error at each time step that lends itself to a more graphical interpretation. The *local error* is the amount by which the numerical solution after one step differs from the value of the true solution to the ODE that passes through the previous numerical solution $\mathbf{y}_{n-1}$. Figure 8 illustrates this error.

As an example, we shall determine the local error of the implicit Euler method. Let $\mathbf{u}(t)$ be the analytic solution to the initial value problem

$$\mathbf{u}'(t)=\mathbf{f}[t,\mathbf{u}(t)],\quad \mathbf{u}(t_{n-1})=\mathbf{y}_{n-1},$$

where $\mathbf{y}_{n-1}$ is the value of the numerical solution at $t_{n-1}$. Applying one step of the method, we obtain

$$\mathbf{u}_n=\mathbf{y}_{n-1}+h_n\mathbf{f}(t_n,\mathbf{u}_n).$$

The local error is given by

$$\mathbf{l}_n=\mathbf{u}_n-\mathbf{u}(t_n).$$

From $\mathbf{d}_n=\mathbf{y}_{n-1}+h\mathbf{f}[t_n,\mathbf{u}(t_n)]-\mathbf{u}(t_n)$, we find that

$$\mathbf{l}_n=\left(I-h\frac{\partial\mathbf{f}}{\partial\mathbf{y}}\right)^{-1}\mathbf{d}_n+O(h^3).$$

If the implicit Euler method is applied to nonstiff systems, the local error and local truncation error are nearly the same, whereas for stiff systems, where $h\partial\mathbf{f}/\partial\mathbf{y}$ is large, these two measures of the error are quite different. However, both are $O(h^2)$ in the limit of small $h$.

There is yet another measure of the error that is, in a sense, the most relevant for the user of ODE and DAE codes. This is global error, which we touched on briefly. The global error is the difference between the numerical

solution and the true solution to the initial value problem. In the case of either of the two Euler methods, the fact that $\mathbf{d}_n = O(h^2)$ can be used to prove that the global errors $\mathbf{y}_n - \mathbf{y}(t_n)$ are $O(h)$.

One might ask, why bother with the local error and the local truncation errors? The reason is that most ODE codes do not attempt to estimate or control the global error because it is very expensive to do so. Instead, they typically estimate either the local error or the local truncation error, and attempt to control the step size so that a norm of this error is smaller than a user-selected error tolerance. The global error is the result of the propagation of local errors over many time steps. Its eventual size depends not only on the size of the local errors, but on the stability of the method and of the differential equation as well. Local error control in a code can be viewed as a knob that can be turned to try to adjust the step sizes and hence the global error. It is not a guarantee of a small global error.

Finally, we have touched on the notion of an *error estimate*. This is the difference approximation that a code makes to estimate the dominant term of the local truncation error or the local error. For the implicit Euler method, the local truncation error depends on the local value of $\mathbf{y}''$. This second derivative can be approximated by difference in $\mathbf{y}$ over the past three points: $t_{n-2}$, $t_{n-1}$, and $t_n$. Equivalently, it is approximately proportional to the difference between the computed value $\mathbf{y}_n$ and the explicit Euler prediction of $\mathbf{y}_n$. This type of difference approximation of the leading term of the local truncation error is often used in codes based on multistep methods (described) because the predictor and corrector values in Part II are readily available.

Another type of error estimate is one obtained by computing the solution by two different methods, one of which is locally more accurate than the other. The difference between the locally computed solutions is an approximation to the error of the less accurate method. This type of error estimate is often used in codes based on Runge–Kutta methods, which do not keep past solution values.

Finally, another way to obtain an error estimate is to compute the solution with two different step sizes and to compute the estimate on the basis of its known asymptotic behavior as $h \to 0$. This type of error estimate is often used in codes based on extrapolation methods.

All of these error estimates are valid in various somewhat idealized situations. It is important to understand, however, that nearly all codes estimate the local error or the local truncation error, and not the global error.