Chapter 5

# Numerical Methods

## 5.1. Introduction

In the previous chapters we have developed a theoretical understanding of initial value problems for ODEs. Only rarely can these problems be solved in closed form, and even when closed-form solutions do exist, their behavior may still be difficult to understand. To gain greater insight, solutions are most commonly approximated numerically using discretization methods. This chapter is intended as a survey of these methods focusing on how they are designed, implemented, and analyzed.

Numerical methods are designed to approximate solutions of locally well-posed initial value problems

$$\mathbf{y}' = f(t, \mathbf{y}), \quad \mathbf{y}(t_o) = \mathbf{y}_o, \quad \mathbf{y} \in \mathbf{R}^d. \tag{5.1}$$

Well-posedness means that there exists a unique solution $\mathbf{y}(t; t_o, \mathbf{y}_o)$ that satisfies (5.1) on a maximal interval of existence $[t_o, t_o + T_*)$, $0 < T_* \leq +\infty$, and that depends continuously on $(t_o, \mathbf{y}_o) \in \mathbf{R}^{d+1}$. We will assume that $\mathbf{f}(t, \mathbf{y})$ is continuous in its first argument, $t$, and locally uniformly Lipschitz continuous in its second argument, $\mathbf{y}$, i.e.,

$$||\mathbf{f}(t, \mathbf{y}_1) - \mathbf{f}(t, \mathbf{y}_2)|| \leq L||\mathbf{y}_1 - \mathbf{y}_2||, \tag{5.2}$$

for some $L > 0$ and any $\mathbf{y}_1, \mathbf{y}_2$ in a neighborhood of $\mathbf{y}_o$. As we have shown in previous chapters, these assumptions guarantee local well-posedness.

Discretization methods employ approximations of (5.1) to construct a discrete set of $\mathbf{y}$-values, $\mathbf{y}_n$, $n = 0, 1, \ldots$, in such a manner that $\mathbf{y}_n$ should approximate $\mathbf{y}(t_n)$ at a corresponding set of $t$-values, $t_n$, called *time-steps*, as the separation of the time-steps, $h_n = t_{n+1} - t_n$, tends uniformly to zero. For most purposes, we will restrict ourselves to $h_n$ that do not vary with $n$ and call their common value $h = t_{n+1} - t_n$ the *step-size* or *discretization parameter*. Variable step-size methods are also useful, but we will only discuss them briefly in the context of automatic error control. We are often interested in the behavior of a discretization method as the discretization parameter decreases to zero, in which case the meaning of $\mathbf{y}_n$ becomes ambiguous. When it is required for clarity in such situations, we will write $\mathbf{y}_{n,h}$ to indicate both the step number and the step-size and otherwise suppress the explicit dependence on $h$.

Discretization methods are broadly categorized as explicit or implicit. Briefly, an *explicit method* obtains the successive values of $y_{n+1}$ parametrically in terms of given or previously computed quantities and is represented symbolically in the form

$$\mathbf{y}_{n+1} = \mathbf{H}(\mathbf{f}, t_n, \ldots, t_{n+1-m}, \mathbf{y}_n, \ldots, \mathbf{y}_{n+1-m}).$$

In contrast, an *implicit method* defines $\mathbf{y}_{n+1}$ as the solution of an equation:

$$\mathbf{G}(\mathbf{f}, t_{n+1}, \ldots, t_{n+1-m}, \mathbf{y}_{n+1}, \ldots, \mathbf{y}_{n+1-m}) = 0$$

that cannot in general be put in the explicit form above.

Discretization methods are also characterized by the number $m$ of previously computed quantities, or *steps*, that the method uses to compute each subsequent approximate value of the solution and by the number of evaluations of the vector field $\mathbf{f}$, or *stages*, that are used per time-step. In the next section, we introduce some basic examples that illustrate the considerations involved in choosing between explicit or implicit methods, single- or multistep, and one- or multistage methods, in order to obtain the greatest computational efficiency in different situations.

All of the *$r$-stage one-step methods* we will consider can be written in the form that characterizes methods known as *Runge-Kutta*

*Methods.* Given a numerical initial value $y_0$, these methods take the specific form

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{r} \gamma_i \mathbf{y'}_{n,i}, \quad n = 0, 1, \ldots \tag{5.3}$$

where

$$\mathbf{y'}_{n,i} = \mathbf{f}(t_n + \alpha_i h, \mathbf{y}_n + h\sum_{j=1}^{r} \beta_{ij}\mathbf{y'}_{n,j}) \quad \text{and} \quad \alpha_i = \sum_{j=1}^{r} \beta_{ij}. \tag{5.3$'$}$$

If $\beta_{ij} = 0$ for $j \geq i$, the method is explicit; otherwise it is implicit. The strategy behind these methods is to obtain better approximations of $\mathbf{y}(t_{n+1})$ by sampling the vector field $\mathbf{f}(t, \mathbf{y})$ at $r$ points near the solution curve emanating from $(t_n, \mathbf{y}_n)$. Each additional sample provides cumulatively better estimates of the solution curve, and thus subsequent samples can also be chosen more usefully. The analytical initial value is sufficient to initialize a one-step method, and no storage of previously computed values is required.

All of the $m$-step one-stage methods we will consider can be written in the form that characterizes methods known as *linear $m$-step methods*. Given numerical initial values $\mathbf{y}_0, \ldots, \mathbf{y}_{m-1}$, these methods take the specific form

$$\mathbf{y}_{n+1} = \sum_{j=0}^{m-1} a_j \mathbf{y}_{n-j} + h\sum_{j=-1}^{m-1} b_j \mathbf{y}'_{n-j}, \qquad n = 0, 1, \ldots, \text{ where } \mathbf{y}'_j = \mathbf{f}(t_j, \mathbf{y}_j). \tag{5.4}$$

If $b_{-1} = 0$, the method is explicit; otherwise $\mathbf{y}_{n+1}$ appears on the right-hand side in the form $\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})$ and the method is implicit. The strategy behind these methods is to obtain better approximations of $\mathbf{y}(t_{n+1})$ by using information from $m$ prior approximations and vector field evaluations, $t_j$,$\mathbf{y}_j$,$\mathbf{f}(t_j, \mathbf{y}_j)$, $j = n, \ldots, n-(m-1)$ that have been stored or generated for initialization. In contrast to multistage methods, only one evaluation of the vector field $\mathbf{f}$ defining the ODE is required per time-step. Discussions of more general methods that combine both Runge-Kutta and multistep characteristics can be found in [GCW] and other references listed in the Web Companion.

Even a discussion of numerical methods must address "theory" as well as "practice". First and foremost, one needs to answer the theoretical question of whether the values obtained by applying a method converge to the analytical solution $\mathbf{y}(t_o+T)$, as the discretization parameter tends to zero and the number of steps increases in such a way that the time interval they represent remains fixed. We call a method *convergent* if and only if, for any IVP (5.1) satisfying (5.2) and any $T > 0$ such that $t_o + T \in [t_o, t_o + T_*)$, the values $\mathbf{y}_{n,h}$ obtained from the method satisfy

$$||\mathbf{y}(t_o + T) - \mathbf{y}_{n,h}|| \to 0 \tag{5.5}$$

as $n \to \infty$ and $h = T/n$. Note that (5.5) implies that $y_{n,h}$ exists for sufficiently large $n$, an issue for implicit methods. If a method is explicit, $\mathbf{y}_{n,h}$ is defined for any $h > 0$ and $n > 0$. The existence of $\mathbf{y}_{n,h}$ is only an issue for implicit methods since they are defined for any $h > 0$ and $n > 0$ if the method is explicit.

We will analyze both the theoretical convergence and practical efficiency of a numerical method in terms of two essential concepts, accuracy and absolute stability. The order of accuracy of a (convergent) method refers to how rapidly errors decrease in the limit as the step-size tends to zero. We say that such a method converges with *order of accuracy* $P$, or, simply, is a *$P$th-order accurate* method, if and only if there exists a $C > 0$ depending only on $\mathbf{y}$, its derivatives, and $T$, such that

$$||\mathbf{y}(t_o + T) - \mathbf{y}_{n,h}|| \leq Ch^P = C\left(\frac{T}{N}\right)^P \tag{5.6}$$

as $n \to \infty$ and no such estimate holds for any greater value of $P$. The dependence of $C$ on $T$ can be removed by considering closed subintervals of the maximal interval of existence. The potential significance of accuracy is immediate: if the increase in computational effort per step required to achieve higher-order accuracy is outweighed by reducing the number of steps required to obtain an approximation within a desired tolerance, the overall computation can be performed more efficiently.

Different notions of stability for numerical methods refer to its tendency 1) to dissipate, 2) to not amplify, or 3) to not uncontrollably amplify perturbations introduced into an approximation. It is well known that there is conflicting nomenclature for certain numerical methods. Less well known is the fact that the term used to describe one of the most essential characteristics of a numerical method, its *absolute stability*, is defined by property 1) in some treatments but by property 2) in others! (See the Web Companion for some examples.) Both properties rule out unbounded growth of perturbations when applied to a problem or class of problems using a particular time-step, so that any systematic amplification is prohibited. Because we wish to encompass the two main types of well-posed initial value problems, those modeling oscillation, transport, and waves, along with those modeling dissipation and diffusion, we shall follow the convention that says that a method is absolutely stable with respect to a particular ODE and step-size $h$ if the numerical solution is bounded as $t_n \to \infty$. Specifically, there exists a $C > 0$ depending only on the initial value such that

$$||\mathbf{y}_{n,h}|| \leq C \tag{5.7}$$

for all $n \geq 0$. There are also reputable treatments whose definition of absolute stability also requires that $||\mathbf{y}_{n,h}|| \to 0$ as $t_n \to \infty$.

For the theoretical considerations of convergence, only the much weaker notion of stability corresponding to property 3) is a necessary condition. This minimal form of stability is called either 0-*stability*, or just plain *stability*. A method is 0-stable with respect to a given problem if, for sufficiently small $h$, the growth of perturbations introduced in each step, representing errors made in prior approximations, can be controlled by some (possibly growing and problem-dependent) function of time. Formally, we say that a method is 0-stable with respect to a particular ODE (5.1), (5.2) if there exists a step-size $h_o > 0$ such that for any $N$ making $0 < h = T/N < h_o$, and all $0 \leq n \leq N$, the difference between the numerical solution $\mathbf{y}_{n,h}$ and any numerical solution $\mathbf{y}_{n,h;\delta}$, defined by the same method with the same step-size, but with perturbations of magnitude no greater than $\delta > 0$ introduced initially and added to the resulting, cumulatively

perturbed values $\mathbf{y}_{n+1,h;\delta}$ at each subsequent step, satisfies

$$||\mathbf{y}_{n,h} - \mathbf{y}_{n,h;\delta}|| \leq C(T)\delta. \tag{5.8}$$

for some positive function of the interval length, $C(T)$. Calling such a method 'not uncontrollably unstable' might be more appropriate. We might wonder if all approximation methods that formally approximate (5.1) satisfy this condition. Indeed, the Runge-Kutta Methods (5.3) are 0-stable by construction. However, we will see examples of linear multistep methods that arise from approximations that are formally quite accurate but that violate 0-stability. A method that is not 0-stable and thus nonconvergent should only be considered in order to understand the causes of its failure, never for actual computation. The facts that 0-stability only involves sufficiently small step-sizes and that it is associated with absolute stability for the problem $y' = 0$, or equivalently for the step-size $h = 0$, explain the terminology.

We can now describe how we aim to understand the theoretical behavior, convergence and higher-order convergence, of these classes of numerical methods. Various equivalent conditions characterize the order of accuracy a method will attain *if* it is also 0-stable. The simplest example of such a condition is based on applying a method to the scalar ODE $y' = 1$, $y(t_o) = y_o$ (using exact initialization $y_j = y_o + (t_j - t_o)$, $j = 0, \ldots, m-1$, in the multistep case). If the resulting solution is exact at all subsequent time-steps, $y_n = y_o + (t_n - t_o)$ for all $n > 0$, we say that the method is *consistent*. A fundamental result is that 0-stability and consistency are not only necessary, but together they are sufficient for a method to be convergent with order $P = 1$. Higher-order potential (or formal) accuracy of a linear multistep method, subject to 0-stability, is equivalent to exactness on ODEs whose solutions are polynomials of degree $\leq P$. For each successive order, this corresponds to one condition that can either be expressed as a linear equation in the coefficients in (5.4), in terms of the asymptotic behavior of a certain polynomial formed from these coefficients, or in terms of dependence on $h$ of the magnitude of errors the method introduces at each step. And while we have noted that 0-stability is inherent in the form of Runge-Kutta Methods, the conditions on their coefficients required for each additional degree of

accuracy are nonlinear and their number grows exponentially, while the number of coefficients only grows quadratically with the number of stages.

Convergence is not a guarantee that a method will perform even adequately in practice. The error bound that guarantees convergence and describes its rate includes contributions reflecting both the accuracy and absolute stability of a method. Growth of perturbations does not prevent high-order convergence, but the growth of perturbations that 0-stability permits can easily dominate these estimates until the step-size $h$ is far smaller than accuracy considerations alone would otherwise require, and so render high-order accuracy irrelevant. For this reason, absolute stability for moderate step-sizes with respect to the problem at hand is the more practically significant property. So no matter how high the order of convergence that theory predicts, we will see that absolute stability analysis is often the deciding factor in performance. There are two situations in particular where these phenomena are especially important. One occurs in the case of multistep methods, whose solutions depend on initial conditions not present in the analytical problem. Because of this, multistep methods have more modes of potential amplification of perturbations than one-step methods. The other occurs in systems whose modes have widely separated temporal scales and applies to both one-step and multistep methods. This separation is quantified through a ratio of magnitudes of eigenvalues of the linearized system. As we shall see, such modes inevitably appear and are excited when we attempt to use discretized systems to accurately approximate the behavior of interest in certain important systems of ODE. Such systems are usually referred to as being *stiff* in the numerics literature. Absolute stability of a method with respect to those peripheral modes and step-size used ensures that small errors arising in these modes do **not** amplify and overwhelm phenomena in modes we wish to and can otherwise resolve.

In the next section, we will introduce seven basic numerical methods in order to motivate and illustrate the theory, and the role of absolute stability in making the theory useful in practice. These

methods have been chosen as prototypes for multiple reasons. First, they exemplify a number of the important different features and properties numerical methods can possess. Second, their derivations are motivated by approximation techniques that can be generalized to obtain entire families of methods with useful characteristics. And third, they are simple enough that their behavior on two important classes of model problems can be rigorously analyzed and completely understood.

The first class of model problems is the $P$th-order accuracy model problem, $M_A(P)$:

$$y' = f(t), \quad y(0) = y_o, \qquad (M_A(P))$$

where $f(t)$ is any polynomial of degree $\leq P-1$, so that the analytical solution $y(t)$ is a polynomial of degree $P$ satisfying the initial condition. This class of model problems can be used to understand the order accuracy of any linear multistep method, and explicit Runge-Kutta Methods for $P \leq 2$. Exact solutions of this model problem for comparison with numerical solutions are easily obtained by antidifferentiation. For each example method, we will obtain an explicit formula for the approximations $y_n$ that it generates when applied to $(M_A(P))$ for some appropriate degree $P$.

The second class of model problems is the absolute stability model problem, $(M_S(\lambda)$:

$$y' = \lambda y, \quad y(0) = 1, \qquad (M_S(\lambda))$$

where we call $\lambda$ the *model parameter*. We will eventually allow both $\lambda$ and $y(t)$ to be complex, but to begin with, we will take both to be real scalars. These model problems can be used to understand the stability properties, of a method, especially absolute stability, which is why we refer to it as $(M_S(\lambda))$. For these homogeneous, first-order, linear, constant coefficient, scalar ODEs, amplification of perturbations amounts to the same thing as amplification of solutions, and therefore absolute stability with respect to $(M_S(\lambda))$, sometimes called linearized absolute stability, forbids unbounded growth of the numerical solutions themselves. For Runge-Kutta Methods this is equivalent to restricting $w = \lambda h$ so that the *amplification factor* of the method $a(w) = y_{n+1}/y_n$ satisfies $|a(w)| \leq 1$. (As noted above, some authors require $|a(w)| < 1$.) We call the set $\{w \in \mathbf{C} \mid |a(w)| \leq 1\}$ the

*region of absolute stability* of the method. The set $\{w \in \mathbf{C} \mid |a(w)| < 1\}$ has been called the linearized stability domain, or lsd [IA1, p. 68]. Though it may seem surprising, we will also consider the analytical solution, $y_{n+1} = e^{\lambda h} y_n$ as a numerical method. We will when we solve systems of ODEs designed to approximate constant coefficient PDEs, where it is known as the Fourier, or *spectral*, method. The spectral method is often used in conjunction with nonexact methods via a technique known as splitting. The region of absolute stability of the analytical solution method is $\{w \in \mathbf{C} \mid \mathrm{Re}(w) \leq 0\}$. We are primarily interested in the absolute stability of other methods for these values of $w$ although nonlinear stabilization can lead us to consider problems with $\mathrm{Re}(w) > 0$ as well.

The model problems $(M_S(\lambda))$ are universal, in the sense that their solutions form a basis for the space of homogeneous solutions of any diagonalizable systems of first-order constant coefficient ODEs. Such systems arise upon linearizing more general systems of ODEs about an equilibrium. They also appear in the spatial discretizations of PDEs of evolution discussed in Section 5.6. The absolute stability model problems for negative real $\lambda$ arise directly from eigenfunction expansions of solutions of the diffusion equations mentioned above, and for purely imaginary $\lambda$ in expansions of solutions of wave equations. The model parameter corresponds to the characteristic exponent (eigenvalue) of a mode. Our analysis will demonstrate rather extreme consequences when numerical methods lack absolute stability with respect to the approximating ODEs and the step-size employed.

We shall invite the reader to implement each example method on $(M_S(\lambda))$ and experiment with its behavior for certain combinations of $\lambda$ and $h$. We can use these programs to perform accuracy studies of each method by fixing $\lambda = -1$ while we successively halve $h$, keeping $Nh = T$, and absolute stability studies by fixing $h$ and successively doubling $\lambda$. To understand our results, we will describe the amplification factor and region of absolute stability for each method.

The results regarding theoretical convergence can be formulated quite nicely in terms of the model problems as follows. A method is 0-stable if and only if it is absolutely stable applied to $M_S(0)$, a condition that is automatically satisfied by Runge-Kutta Methods.

We say that a numerical method has *formal accuracy* (or *polynomial accuracy*) of order $P$ if it can be applied to every problem in the class $(M_A(P))$ using exact initial values and the resulting numerical solution is exact ($y_n = y(t_n)$ for all time-steps $t_n$, $n \geq 0$). Because of this, a method is consistent if and only if it has polynomial accuracy of order $P \geq 1$. Therefore a method is convergent if and only if it is exact in the sense above on $M_A(1)$ and absolutely stable on $M_S(0)$. In terms of its coefficients, the Runge-Kutta Method (5.3), (5.3′) is consistent if and only if $\sum_{i=1}^{r} \gamma_i = 1$, and the the linear multistep method (5.4) is consistent if and only if $\sum_{j=0}^{m} a_j = 1$ and $\sum_{j=-1}^{m} b_j - \sum_{j=0}^{m} j a_j = 1$.

Each additional degree of polynomial accuracy depends on one additional algebraic condition on the coefficients. Each degree of formal, polynomial accuracy implies another order of actual accuracy for 0-stable linear multistep methods. For Runge-Kutta Methods, beyond second order, this polynomial accuracy of degree $P$ is insufficient to guarantee general accuracy. For $P = 3$, an additional condition on the accuracy of a Runge-Kutta Method when applied to $M_S(P)$ is sufficient to guarantee general $P$th-order accuracy, but even this is insufficient for any greater $P$. For this reason, a more correct name for the model problems $(M_A(P))$ might be the linear multistep accuracy model problems, and for $P = 1$, the consistency model problem. Further details and discussion of issues pertaining individually to Runge-Kutta Methods and linear multistep methods may be found in Appendices H and I, respectively.

Although the model problems $M_S(0)$ and $M_A(0)$ both refer to the same ODE, $y' = 0$, 0-stability only excludes unbounded growth; it does not require exactness. Numerical solutions of $M_S(\lambda)$ obtained from the linear multistep method (5.4) remain bounded as $n \to \infty$ if and only if the roots of the characteristic polynomial of the method

$$p_w(r) = \rho(r) - w\sigma(r), \text{ where}$$
$$\rho(r) = r^m - \sum_{j=0}^{m-1} a_j r^{m-(j+1)}, \text{ and } \sigma(r) = \sum_{j=-1}^{m-1} b_j r^{m-(j+1)} \tag{5.9}$$

satisfy the following root condition: All roots are either inside the unit

circle in the complex plane or on the unit circle and simple. In fact, one definition of absolute stability for a linear multistep method with respect to $M_S(\lambda)$ is that the roots of $p_w(r)$ satisfy the root condition when $w = \lambda h$. Under that definition of absolute stability, 0-stability is immediately equivalent to absolute stability with respect to $M_S(0)$, and to the fact that $\rho(r)$ satisfies the root condition.

So for methods of the form (5.3), (5.3′), and (5.4) the classes of model problems $M_A(P)$ and $M_S(\lambda)$ are sufficient to determine the theoretical issue of convergence, the practical issue of linearized absolute stability, and even higher-order accuracy, except for Runge-Kutta Methods beyond $P = 3$. But even for linear multistep methods, we will discover that attaining high-order accuracy tends to compete with retaining absolute stability when $h$ is not vanishingly small. Only by increasing the per-step computational effort, including the number of evaluations of the vector field required for each time-step can both be increased independently. In particular, implicit methods that permit greater accuracy without compromising stability require additional effort to solve the nonlinear equations that define each step.

In conclusion, many factors must be considered in choosing or designing an efficient method for a specific problem and its parameters. Our understanding of the practical performance of numerical methods can be guided effectively in terms of the order of accuracy and absolute stability of a method. These two concepts interact to determine the step-size required to obtain an approximation within a specified tolerance, when using a particular method on a particular ODE. Along with the problem-specific computational effort required per step, reflecting the number of stages and steps in the method and the work required to evaluate the vector field, they determine the relative efficiency of different methods in different situations. It is important to realize that no method is universally superior to all others, and the selection of an effective method depends upon careful consideration of features of the problem or class of problems one wishes to solve and the accuracy of approximation required.

## 5.2. Fundamental Examples and Their Behavior

Now we introduce several working examples of numerical methods for IVPs that are motivated by relatively elementary principles. Then we will apply them to the model problems we introduced above. We will focus on how their behavior depends on the nature of the problem and the step-size. We do not claim that a method that performs well on such simple problems will necessarily perform well on more challenging problems. However, methods that perform poorly on simple problems with certain features will likely *not* perform well on more complex problems with similar features.

• **Example 5–1. Euler's Method.** The most familiar and elementary method for approximating solutions of an initial value problem is Euler's Method. Euler's Method approximates the derivative in (5.1) by a finite difference quotient $\mathbf{y}'(t) \approx (\mathbf{y}(t+h) - \mathbf{y}(t))/h$. We shall usually discretize the independent variable in equal increments:

$$t_{n+1} = t_n + h, \quad n = 0, 1, \ldots, t_0 = t_o. \tag{5.10}$$

Henceforth we focus on the scalar case, $N = 1$. Rearranging the difference quotient gives us the corresponding approximate values of the dependent variable:

$$y_{n+1} = y_n + hf(t_n, y_n), \quad n = 0, 1, \ldots, y_0 = y_o. \tag{5.11}$$

Euler's Method is an $r$-stage Runge-Kutta Method (5.3) with $r = 1$, $\gamma_1 = 1$, and $\beta_{11} = 0$. It is also a linear $m$-step method (5.4) with $m = 1$, $a_0 = 1$, $b_{-1} = 0$, and $b_0 = 1$. Since $b_{-1} = 0$, it is explicit. However, it is too simple to capture essential features that occur for $m$ or $r > 1$ and that we will find present in our next examples.

Geometrically, Euler's Method follows the tangent line approximation through the point $(t_n, y_n)$ for a short time interval, $h$, and then computes and follows the tangent line through $(t_{n+1}, y_{n+1})$, and so on, as shown in Figure 5.1.

▷ **Exercise 5–1.** Write a program that implements Euler's Method, the values of $f(t, y)$ coming from a function defined in the program. Test the results on the model problem $(M_S(\lambda))$,

$$y' = \lambda y, \quad y(0) = 1, \quad t \in [0, T], \quad y, \lambda \in \mathbf{R},$$

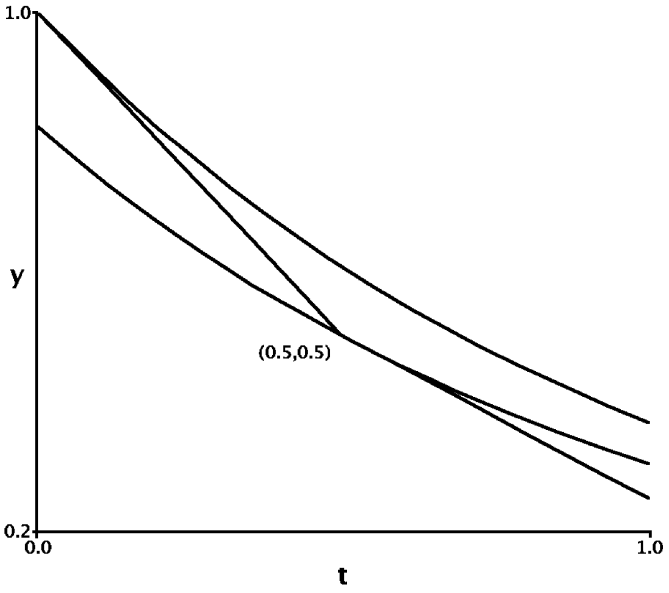


Figure 5.1. Interpretation of Euler's Method.

whose analytical solution is $y(t) = 1e^{\lambda t}$. Use $T = 1$ and combinations of $\lambda = \pm 2^l, l = 0, 1, \ldots, L$, and $h = 2^{-m}, m = 0, 1, 2, \ldots, M$, $L = 5$, $M = 5$.

The anticipated results for $l = 0$ ($\lambda = -1$) and $m = 0, \ldots, 4$ are displayed along with the exact solution in Figure 5.2.

Strictly speaking, Euler's Method generates a sequence of points in the $(t, y)$-plane, but we conventionally associate this sequence with the piecewise-linear curve obtained by joining consecutive points with line segments. Figure 5.2 shows three of these approximating curves and illustrates an important distinction involved in analyzing the convergence of these approximations. We call the difference between the exact solution and an approximate solution at a certain value of $t_o + T$ a *global error*, since it is the cumulative result of local errors introduced in each of $N$ steps of size $h = T/N$ and the propagation of errors accumulated in earlier steps to later steps. These errors may either be amplified or attenuated from earlier steps to later steps.

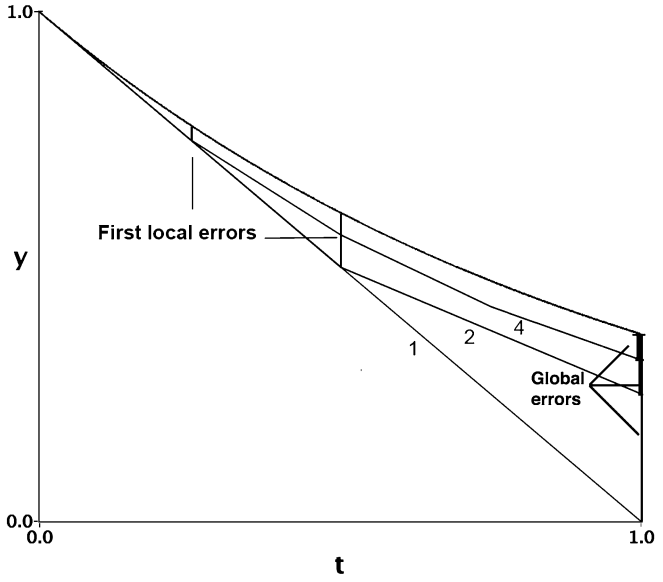


Figure 5.2. Behavior of Euler's Method: Accuracy.

Global errors corresponding to $N = 1, 2$, and 4 are represented by vertical lines at $T = 1$. As this implies, a *local error* does not include the effect of prior errors but is the difference between one step of an approximate solution and the exact solution sharing the same initial value and time interval over that one step. Local errors for one step of Euler's Method starting at $(0, 1)$, with $N = 1, 2$, and 4, are represented by vertical lines at $T = 1$, $1/2$, and $1/4$, respectively. Two kinds of local errors are discussed and depicted in the final section of the chapter on convergence analysis. Local truncation errors arise when a step of a method is initialized using the exact solution values. Another local error involves local solutions $\hat{y}_n(t)$ passing through values of the numerical solution $(t_n, y_n)$. In Figure 5.2, $h$ decreases by factors of $1/2$ while the number of steps doubles. The figure indicates that local errors for Euler's Method are on the order of $h^2$. At $t_N = T = 1$ it appears that the difference between the approximate solution and the analytical solution also decreases by a factor of $1/2$.

This suggests that for Euler's Method, global errors are on the order of $h$.

According to the description in the introduction, the near proportionality of errors at $y(1)$ to $h^1$ suggests that Euler's Method has order of accuracy 1, or in other words that it is *first-order accurate*.

To perform an analytical *accuracy* study of Euler's Method, we apply it to the class of accuracy model problems ($M_A^2$) written in the form

$$y' = \frac{d}{dt}(c_1(t-t_o) + c_2(t-t_o)^2), \quad y(t_o) = y_o, \tag{5.12}$$

whose analytical solution is

$$y(t) = y_o + c_1(t-t_o) + c_2(t-t_o)^2. \tag{5.12'}$$

When Euler's Method is applied to (5.12), it reduces to $y_{n+1} = y_n + h(c_1 + 2c_2nh)$. Using $\sum_{n=1}^{N-1} 2n = N(N-1)$, we find that $y_N = y_0 + c_1Nh + c_2h^2(N^2 - N)$. In terms of $t_n - t_o$,

$$y_N = y_0 + c_1(t_n - t_o) + c_2(t_n - t_o)^2 - c_2h(t_n - t_o)$$

and the global error at time $T = Nh$ satisfies

$$y(t_o + T) - y_N = (y_o - y_0) + c_2Th.$$

Setting $c_2 = 0$ shows that Euler's Method is exact when $y(t)$ is a polynomial of degree 1. For a polynomial of degree 2, its error at a fixed $T$ is proportional to the first power of the time-step $h$. When we estimate the global error in the general case to prove convergence, the bound will involve a factor $\max_{t\in[t_o,t_o+T]} \frac{y''(t)}{2}$ that reduces to the factor of $c_2$ above.

To perform an analytical *absolute stability* study of Euler's Method, we apply it to the class of stability model problems ($M_S(\lambda)$). When Euler's Method is applied to these model problems, it reduces to $y_{n+1} = (1+w)y_n$, where we have combined the model parameter $\lambda$ and discretization parameter $h$ into a single parameter $w = \lambda h$. The exact solution with the same initial condition $y_n$ and time interval $h$ is $y_ne^{\lambda h} = y_n(1 + \lambda h + (\lambda h)^2/2 + (\lambda h)^3/3! + \cdots)$. In this context, one step of Euler's Method captures the terms of order $h^1$ in the exact solution correctly, and the remainder is bounded by a multiple

of $h^2$. Successive iterates of Euler's Method can be written as $y_n = (1+w)^n y_o$. Heuristically, $N$ errors of order $h^2$ accumulate to give an error of order $h$. Stability is necessary in order to make this argument rigorous.

The absolute stability properties of Euler's Method are illustrated by results of the exercise for $m = 3$ ($h = 1/8$) and $\lambda = -2^l$, $l = 0, \ldots, 5$, displayed in Figure 5.3. The value of $\lambda$ appears adjacent to the corresponding exact and approximate solutions. As $\lambda$ grows progressively more negative to $\lambda = -16$ where $w = -2$ and $y_n = (1+w)^n y_o = (-1)^n$, the approximate solution does not decay but simply oscillates. Beyond this value, e.g., $\lambda = -32$ so $w = -4$ and $(1+w)^n = -3^n$, the oscillations grow exponentially as shown in Figure 5.4. Note that some of the approximations in the accuracy and stability figures share the same values of $w$, e.g., $\lambda = -8$, $h = 1/8$, and $\lambda = -1$, $h = 1$; the independent variable must still be rescaled in order to make them correspond exactly. According to the description in the introduction, it follows from the form of the above solution that Euler's Method is absolutely stable with respect to the model problem when $|1 + \lambda h| \leq 1$. We will also be examining the model problem in the complex plane—that is, we will interpret $\lambda$ as complex and replace the real scalar $y$ with its complex equivalent, $z$. Thus we call $\{w \in \mathbf{C} \mid |1+w| \leq 1\}$, the closed disc of radius 1 about the point $w = -1$, the *region of absolute stability* for Euler's Method. The region of absolute stability of Euler's Method is depicted in Figure 5.14, together with the corresponding regions for the remaining example methods after they too have been analyzed.

In some important applications we will consider later, the instability exhibited by Euler's Method in Figure 5.4 has unavoidable negative consequences that can only be resolved by resorting to implicit methods such as those that we will derive below. In other circumstances, the rate of convergence exhibited by Euler's Method in Figure 5.2 is also unsatisfactory, and practical efficiency considerations require methods with higher-order accuracy, obtained from either multistep or multistage strategies. Along with performance improvements, each of these modifications brings with it important implementation considerations that do not appear in Euler's Method

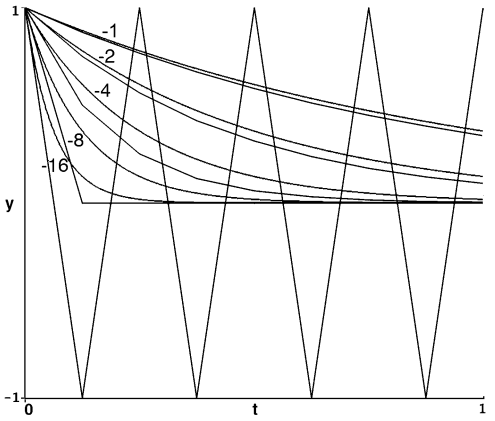


Figure 5.3. Behavior of Euler's Method: Stability.

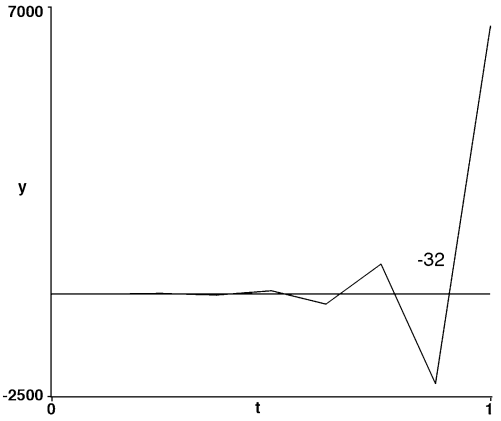


Figure 5.4. Behavior of Euler's Method: Instability.

due to its simplicity. We shall see that these considerations provide both challenges and opportunities.

To obtain the most basic examples of these kinds of methods, we provide another interpretation of Euler's Method and then modify it. In addition to the tangent line and difference quotient interpretations above, Euler's Method can be viewed as arising from the left endpoint

approximation of the integral of $y'$ over an interval of width $h$:

$$y(t+h) - y(t) = \int_t^{t+h} y'(s)\, ds \approx h y'(t). \tag{5.13}$$

To improve upon Euler's Method, we can use more symmetric approximations:

$$\int_t^{t+h} y'(s)\, ds \approx h y'(t + \frac{h}{2}) \tag{5.14}$$

and

$$\int_t^{t+h} y'(s)\, ds \approx h \frac{y'(t) + y'(t+h)}{2}. \tag{5.15}$$

Both are exact if $y$ is a polynomial of degree $\leq 2$ (see Appendix J).

We expect that the methods known as the midpoint method and the leapfrog method, both obtained from (5.14) in the form

$$y(t+h) - y(t) \approx h y'(t + \frac{h}{2}), \tag{5.16}$$

and the trapezoidal method, obtained from (5.15) in the form

$$y(t+h) - y(t) \approx h \frac{y'(t) + y'(t+h)}{2}, \tag{5.17}$$

would lead to more accurate approximations of (5.1) than Euler's Method. In the next section we will show rigorously that they do. The geometric interpretation of these approximations and of the Euler's Method approximation, (5.13), are depicted in Figure 5.5.

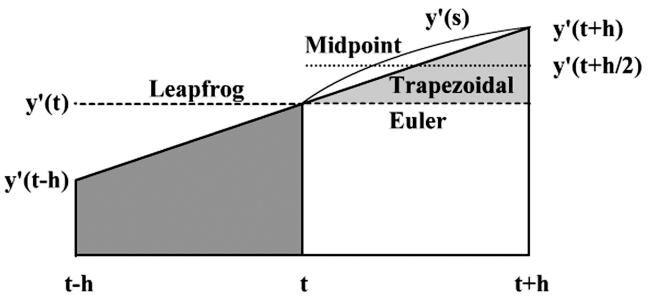


Figure 5.5. Interpretation of some basic methods.

▷ **Exercise 5–2.** The mean value theorem guarantees that the difference quotient $(y(t+h)-y(t))/h$ is equal to $y'(\xi)$, not at $\xi = t$ or $\xi = t+h$, but rather at some point $\xi$ strictly between $t$ and $t+h$. Show that for any polynomial $p_2(t)$ of degree $\leq 2$, (5.14)–(5.17) are exact, that is,

$$\frac{p_2(t+h)-p_2(t)}{h} = p_2'(t+\frac{h}{2})$$

and also

$$\frac{p_2(t+h)-p_2(t)}{h} = \frac{p_2'(t)+p_2'(t+h)}{2}.$$

• **Example 5–2. The midpoint method.** To obtain the midpoint method from (5.16), we discretize $t_n$ as in (5.10) and approximate

$$y'(t+\frac{h}{2}) = f(t+\frac{h}{2}, y(t+\frac{h}{2}))$$

using one step of Euler's Method with time-step $\frac{h}{2}$ as follows:

$$y_{n+\frac{1}{2}} = y_n + hf(t_n, y_n),$$

$$y_{n+1} = y_n + hf(t_n+\frac{h}{2}, y_{n+\frac{1}{2}}), \quad n = 0, 1, \ldots . \tag{5.18}$$

The midpoint method is an explicit $r$-stage Runge-Kutta Method, with $r = 2, \gamma_1 = 0, \gamma_2 = 1$, $\beta_{11} = \beta_{12} = \beta_{22} = 0$, and $\beta_{21} = \frac{1}{2}$.

▷ **Exercise 5–3.** If we accept the fact that the global error in Euler's Method is proportional to $h$ within $O(h^2)$, the midpoint method can be derived using a technique known as *extrapolation*. Show that applying this assumption to one Euler step of size $h$ and two steps of size $\frac{h}{2}$ tells us

$$y_{n+1} = y_n + hf(t_n, y_n) + Ch + O(h^2)$$

and

$$y_{n+1} = y_n + \frac{h}{2}f(t_n, y_n) + \frac{h}{2}f(t_n+\frac{h}{2}, y_n+\frac{h}{2}f(t_n, y_n)) + C\frac{h}{2} + O(h^2).$$

Then form a combination of these formulas, twice the latter minus the former, to obtain the midpoint method:

$$y_{n+1} \approx y_n + hf(t_n+\frac{h}{2}, y_n+\frac{h}{2}f(t_n, y_n)) + O(h^2).$$

▷ **Exercise 5–4.** Modify the program implementing Euler's Method to implement the midpoint method on the model problem ($M_S(\lambda)$), using the same parameters, and compare the results.

For our analytical accuracy study of the midpoint method, we consider the class of initial value problems ($M_A^3$) written in the form

$$y' = \frac{d}{dt}(c_1(t-t_o) + c_2(t-t_o)^2 + c_3(t-t_o)^3), \quad y(t_o) = y_o, \quad (5.19)$$

whose exact solution is

$$y(t) = y_o + c_1(t-t_o) + c_2(t-t_o)^2 + c_3(t-t_o)^3. \quad (5.19')$$

This is simply the next higher-order analogue of (5.12).

If we apply the midpoint method to (5.19), it reduces to

$$y_{n+1} = y_n + h\left(c_1 + c_2(2n+1)h + 3c_3\left(\frac{(2n+1)h}{2}\right)^2\right).$$

Using $\sum_{n=0}^{N-1} 2n+1 = N^2$ and $\sum_{n=0}^{N-1} 3(2n+1)^2 = 4N^3 - N$, we find

$$y_N = y_0 + c_1 Nh + c_2(Nh)^2 + c_3(Nh)^3 - c_3 h^3 \frac{N}{4}.$$

From this, we see that the global error at time $T = Nh$ satisfies $y(t_o + T) - y_N = (y_o - y_0) + \frac{1}{4}c_3 T h^2$. The midpoint method is exact when $y(t)$ is a polynomial of degree 2, and for a polynomial of degree 3, the error at a fixed $T$ is proportional to $h^2$. While formal accuracy analysis using the model problems ($M_A(P)$) does not in general tell the whole story regarding accuracy of Runge-Kutta Methods, if $P \leq 2$, formal order of accuracy $P$ does imply order of accuracy $P$. We will discuss these issues in greater detail below and in Appendix H.

In the context of the model problem, with $w = \lambda h$, the midpoint method becomes $y_{n+1} = (1 + w + w^2/2)y_n$ whose solution is $y_n = (1 + w + w^2/2)^n y_o$.

Figure 5.6 depicts the results of using the midpoint method with the same parameters as in Figure 5.3. As $h$ decreases by factors of $\frac{1}{2}$, the number of steps doubles. At $t_N = T = 1$ it now appears that the difference between the approximate solution and the analytical solution decreases by a factor of 1/4, suggesting that the midpoint

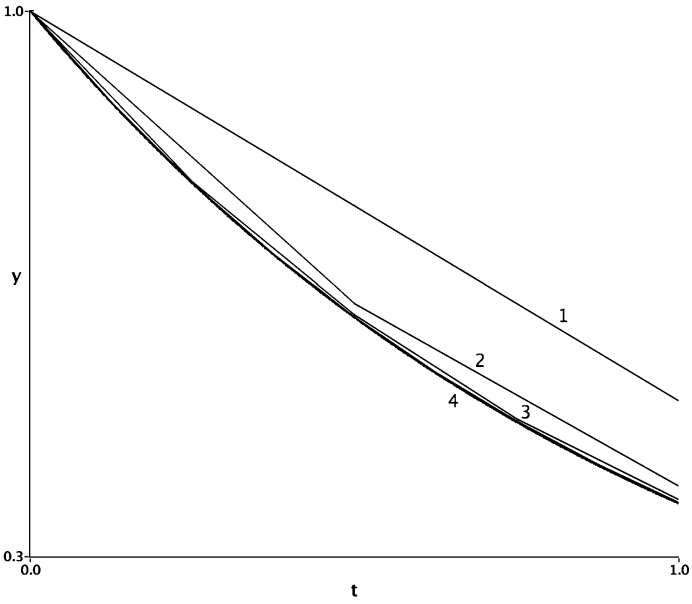


Figure 5.6. Behavior of the midpoint method: Accuracy.

method is second-order accurate; i.e., its order of accuracy is 2. The approximation with $j = 5$, $h = 1/32$ is indistinguishable from the exact solution. We might expect this behavior from the observation that one step of the midpoint method captures the terms of order $\leq w^2$ in the exact solution $y_n e^w = y_n(1 + w + w^2/2 + w^3/3! + \cdots)$ correctly, and the remainder is bounded by a multiple of $w^3$.

In the same fashion, Figure 5.7 corresponds to Figure 5.3, except the final value of $\lambda$ has been changed from $-16$ to $-17$ to illustrate the incipient instability similar to that of Euler's Method when $w < -2$. For $\lambda = -16$, $w = -2$, the approximate solution $y_n = 1^n$ neither decays nor grows, nor does it oscillate as it did when Euler's Method was used. From the solution above, the midpoint method is absolutely stable with respect to the model problem when $|1+\lambda h+(\lambda h)^2/2| \leq 1$. In the complex plane, the region of absolute stability for the midpoint method is then $\{w \in \mathbf{C} \mid |1 + w + w^2/2| \leq 1\}$.

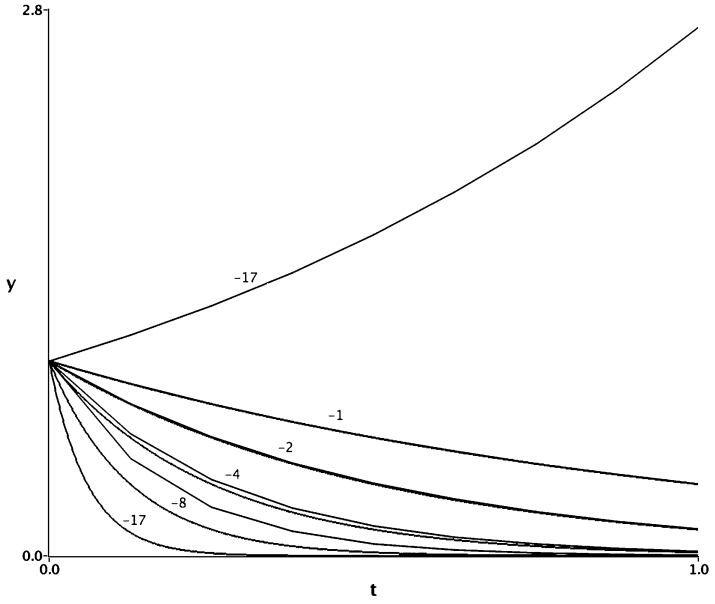


Figure 5.7. Behavior of the midpoint method: Stability.

• **Example 5–3. The Leapfrog Method.** To obtain the leapfrog method, we discretize $t_n$ as in (5.10), but we double the time interval, $h$, and write the midpoint approximation (5.16) in the form

$$y'(t+h) \approx (y(t+2h) - y(t))/h$$

and then discretize it as follows:

$$y_{n+1} = y_{n-1} + 2hf(t_n, y_n). \tag{5.20}$$

The leapfrog method is a linear $m = 2$-step method, with $a_0 = 0$, $a_1 = 1$, $b_{-1} = -1$, $b_0 = 2$, and $b_1 = 0$. It uses slopes evaluated at odd values of $n$ to advance the values at points at even values of $n$, and vice versa, reminiscent of the children's game of the same name. For the same reason, there are multiple solutions of the leapfrog method with the same initial value $y_0 = y_o$. This situation suggests a potential instability present in multistep methods, which must be addressed

when we analyze them—two values, $y_0$ and $y_1$, are required to initialize solutions of (5.20) uniquely, but the analytical problem (5.1) only provides one. Also for this reason, one-step methods are used to initialize multistep methods.

▷ **Exercise 5–5.** Modify the program implementing Euler's Method to implement the leapfrog method on the model problem ($M_S(\lambda)$), using the same parameters. Initialize $y_1$ 1) using the 'constant method', $y_1 = y_0$, 2) using one step of Euler's Method, $y_1 = y_0 + hf(t_0, y_0)$, and 3) using one step of the midpoint method. Compare the results.

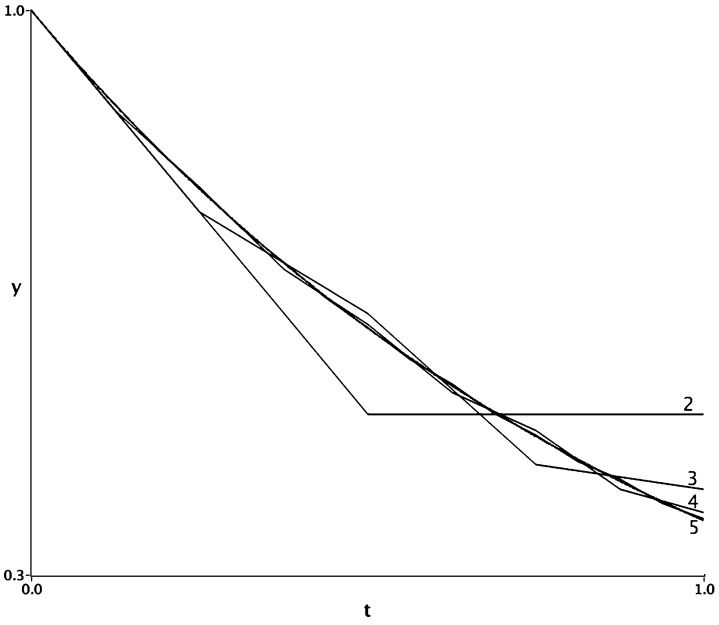


Figure 5.8. Behavior of the leapfrog method: Accuracy.

If we apply the leapfrog method to (5.19), it reduces to

$$y_{n+1} = y_{n-1} + 2h(c_1 + 2c_2nh + 3c_3(nh)^2).$$

If $N = 2K$ is even, we use $\sum_{k=1}^{K} 4(2k-1) = (2K)^2 = N^2$ and $\sum_{k=1}^{K} 6(2k-1)^2 = (2K-1)(2K)(2K+1) = N^3 - N$ to show that

$$y_N = y_0 + c_1 Nh + c_2 (Nh)^2 + c_3 (Nh)^3 - c_3 Nh^3.$$

If $N = 2K+1$ is odd, we use $\sum_{k=1}^{K} 4(2k) = (2K+1)^2 - 1 = N^2 - 1$ and $\sum_{k=1}^{K} 6(2k)^2 = 2K(2K+1)(2K+2) = N^3 - N$ to show

$$y_N = y_1 - (c_1 h + c_2 h^2 + c_3 h^3) + c_1 Nh + c_2 (Nh)^2 + c_3 (Nh)^3 - c_3 (N-1) h^3.$$

From this, we see that the global error at time $T = Nh$ satisfies

$$y(t_o + T) - y_N = (y(0) - y_0) + \frac{1}{4} c_3 T h^2$$

when $N$ is even and

$$y(t_o + T) - y_N = (y(t_1) - y_1) + c_3 (T h^2 - h^3)$$

when $N$ is odd.

Assuming at first that $y_0 = y_o$ and $y_1 = y(t_1)$, the leapfrog method is exact when $y(t)$ is a polynomial of degree 2, and for a polynomial of degree 3, the error at a fixed $T$ is proportional to $h^2$.

When the initial values are not exact, the formulas illustrate the dependence of global errors on the values used to initialize linear multistep methods. If the leapfrog method is initialized with the constant method, i.e., if we use $y_1 = y_o$, then $y(t_1) - y_1 = y(t_1) - y(t_0) = y'(\xi) h$ for some $\xi \in (t_0, t_1)$, the overall error degrades to $O(h)$ and the effort involved in employing a higher-order method is wasted. If we use Euler's Method, $y_1 = y_o + h y'(t_0)$, then $y(t_1) - y_1 = y(t_1) - (y(t_0) + h y'(t_0)) = y''(\xi) \frac{h^2}{2}$ for some $\xi \in (t_0, t_1)$ has the same order of magnitude as the largest other term contributing to the global error. The overall error achieves its maximum potential order of $O(h^2)$. The principle is the same for methods with more steps and initialization values. We only need to initialize using a method whose *global* error has order one less than the method it initializes. Only the local errors of the initialization method affect the global error of the overall method, since they are incurred over a fixed number of steps independent of $h$. In contrast, to reach a fixed $t_o + T$, the number of steps of the method being initialized is $N = T h^{-1}$. This factor is responsible for the different orders of magnitude of global errors, and

both local and initialization errors. There is no benefit gained from any additional effort devoted to computing initialization values more accurately. If we used one step of the midpoint method or Heun's Method instead of Euler's Method to compute $y_1$, improvement in the accuracy of the solution, if any, would be negligible.

We have implicitly assumed that we can use the analytical initial value $y_o$ to initialize a numerical method. But even for a one-step method, sometimes initial values themselves are only computed approximately. If we imagine stopping a computation and then continuing, the results must be identical to those obtained had we not stopped in the first place. If we compare several steps of a computation with the same computation broken into two, the results are clearly the same. In the latter case, the second part begins with an inexact value. Fortunately, the new initial error is the order of the global error of the numerical method, and we have seen that this is all that is required in order for global errors to continue having the same order.

When the leapfrog method is applied to the absolute stability model problem $(M_S(\lambda))$, it takes the form $y_{n+1} = y_{n-1} + 2wy_n$. This is a linear second-order constant coefficient difference equation whose general solution is a linear combination, $y_n = c_+ y_n^+ + c_- y_n^-$, of two basic solutions, $y_n^+ = r_+^n$ and $y_n^- = r_-^n$, where $r_\pm$ are roots of $p_w(r) = r^2 - 2wr - 1$, the *characteristic polynomial* associated with the leapfrog method. In general, we find that $y_j = r^j$ is a nonzero solution of (5.4) if and only if $r$ is a root of the characteristic polynomial of (5.4),

$$p_w(r) = \rho(r) - w\sigma(r)$$
$$\text{where } \rho(r) = r^m - \sum_{j=0}^{m-1} a_j r^{m-(j+1)} \text{ and } \sigma(r) = \sum_{j=-1}^{m-1} b_j r^{m-(j+1)}. \tag{5.21}$$

For any real $w$ the characteristic polynomial of the leapfrog method has two distinct roots given by the quadratic formula, $r_\pm = w \pm \sqrt{w^2+1}$. When $w > 0$, $r_+ > 1$ and $-1 < r_- < 0$, and when $w < 0$, $0 < r_+ < 1$ and $r_- < -1$. (If $w = 0$, then $r_\pm = \pm 1$.)

Therefore, when $\lambda < 0$ and the analytic solution has exponentially decreasing magnitude, the leapfrog method applied to the model problem exhibits unstable exponential growth regardless of how small $h$ may be, as long as $c_- \neq 0$. Since $c_- = 0$ implies $y_{n+1} = r_+ y_n$, if we initialize $y_1$ using one step of Euler's Method, or either of the other methods suggested above, we are guaranteed $c_- \neq 0$. Using a binomial expansion, $(1+u)^{1/2} = 1 + u/2 - u^2/8 + \cdots$, $|u| < 1$, $r_+ = 1 + w + w^2/2 - w^4/8 + \cdots$, $|w| < 1$; i.e., for small $|w|$, one step of the mode $r_+$ of the leapfrog method agrees with the terms of order $\leq w^2$ in the exact solution $y_n e^w$, and the remainder is bounded by a multiple of $w^3$. When $w$ approaches zero along the negative real axis, $r_+ \approx 1 + w$ has magnitude less than 1. Since $r_+ r_- = -1$, or using the expansion above, $r_- \approx -1 + w - w^2/2$, in this situation $r_-$ has magnitude greater than 1 and the powers of $r_-$ explain the exponentially growing oscillations observed in solutions of the leapfrog method.

Figure 5.8 shows a series of results using the leapfrog method with the same parameters as in Figures 5.2 and 5.5—as $h$ decreases by factors of 1/2, the number of steps $N$ gets doubled. (We start with $l = 2$, $h = 1/2$ since it is a 2-step method.) At $t_N = T = 1$, the difference between the approximate solution and the analytical solution decreases by a factor of 1/4, similar to the behavior of the midpoint method. Along with the accuracy model analysis, this adds evidence that the leapfrog method is also second-order accurate. The approximation with $j = 5$, $h = 1/32$ is indistinguishable from the exact solution. Figure 5.9 is a stability study corresponding to to Figures 5.3 and 5.7, but to capture the behavior with different parameters, the time-steps $h$ and time intervals $T$ are varied along with $\lambda$. Starting with $\lambda = -1$ and $h = 1/8$ as before, we have extended the time interval to $T = 8$ to observe the visible onset of instability. Each time $\lambda$ is doubled, we have divided $h$ by four so $w = \lambda h$ is halved, but this only accelerates the onset of instability, and our time intervals must shrink so that further amplification does not prevent us from showing the results on a common graph. From the form of the solution above, the leapfrog method is only absolutely stable with respect to the real scalar model problem when $w = 0$. We will analyze the situation for complex $w$ when we apply the leapfrog method to a

$2 \times 2$ system below. We will see that the region of absolute stability for the leapfrog method is the open interval on the imaginary axis, $\{w \in \mathbf{C} \mid w = bi, -1 < b < +1\}$, i.e., the set of complex $w$ such that $w = -\bar{w}$ and $|w| < 1$. The endpoints are not included because when $w = \pm i$, $p_w(r) = r^2 \pm 2ir - 1 = (r \pm i)^2$ has a multiple root on the unit circle, so the general solution of the difference equation has an algebraically growing mode.

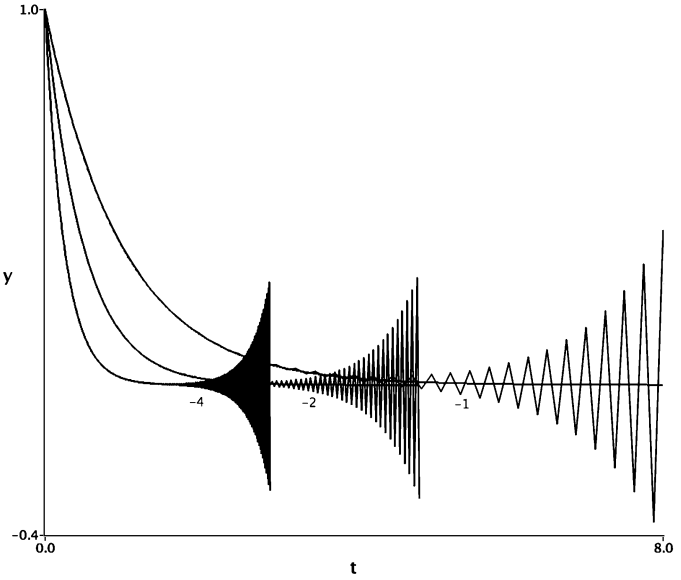


Figure 5.9. Behavior of the leapfrog method: Stability.

• **Example 5–4. The trapezoidal method.** To obtain the trapezoidal method, we define $t_n$ as above and discretize (5.15) as follows:

$$y_{n+1} = y_n + h\frac{f(t_n, y_n) + f(t_{n+1}, y_{n+1})}{2}, \quad n = 0, 1, \ldots . \tag{5.22}$$

The trapezoidal method is classified as an *implicit method* because each step requires solving an equation to obtain $y_{n+1}$. In contrast, Euler's Method is called an *explicit method* because $y_{n+1}$ is given parametrically. We will see that implicit methods can have stability advantages that make them more efficient in spite of this additional computational work needed to implement them.

The trapezoidal method is an implicit linear $m$-step method with $m = 1$, $a_0 = 1$, $b_0 = 1/2$, and $b_{-1} = 1/2$. It is also an implicit $r$-stage Runge-Kutta Method with $r = 2$, $\gamma_1 = \gamma_2 = 1/2$, $\beta_{11} = \beta_{12} = 0$, and $\beta_{21} = \beta_{22} = 1/2$. Even though the trapezoidal method is a 2-*stage* method, only one *new* evaluation of $f$ is required per time-step after the first step. In general, if both $y'_n$ and $y'_{n+1}$ are among the $r > 1$ evaluations of an $r$-stage method, the number of new evaluations per time-step after the first step is $r - 1$.

▷ **Exercise 5–6.** Modify the program implementing Euler's Method to implement the trapezoidal method on the model problem ($M_S(\lambda)$), using the same parameters. To find $y_{n+1}$, you may treat (5.22) as a fixed-point problem $y_{n+1} = g(y_{n+1})$ and implement *fixed-point iteration*, $y_{n+1}^{(k+1)} = g(y_{n+1}^{(k)})$. Or you may rewrite (5.22) in the form $F(y_{n+1}) = 0$ and apply a root-finding method, e.g., *Newton's Method*, $y_{n+1}^{(k+1)} = y_{n+1}^{(k)} - F'(y_{n+1}^{(k)})^{-1} F(y_{n+1}^{(k)})$. In the case of the model problem, using the analytical solution is the simplest approach, although this would not be useful for general ODEs.

If we apply the trapezoidal method to (5.19), it reduces to

$$y_{n+1} = y_n + h(c_1 + c_2(2n+1)h + 3c_3\frac{(nh)^2 + ((n+1)h)^2}{2}).$$

The right-hand side is close to the right-hand side we obtained when we applied the midpoint method to (5.19), only less an additional $3c_3h^3\frac{1}{4}$. Modifying the solution appropriately, we find the trapezoidal method yields

$$y_N = y_0 + c_1Nh + c_2(Nh)^2 + c_3(Nh)^3 - c_3h^3\frac{N}{2}$$

and the global error at time $T = Nh$ satisfies $y(t_o + T) - y_N = (y_o - y_0) + \frac{1}{2}c_3Th^2$. The trapezoidal method is exact when $y(t)$ is a polynomial of degree 2, and for a polynomial of degree 3, the error at a fixed $T$ is proportional to the second power of the time-step, $h^2$.

When the trapezoidal method is applied to the absolute stability model problem ($M_S(\lambda)$), it takes the form

$$y_{n+1} = (1 + w/2)(1 - w/2)^{-1}y_n,$$

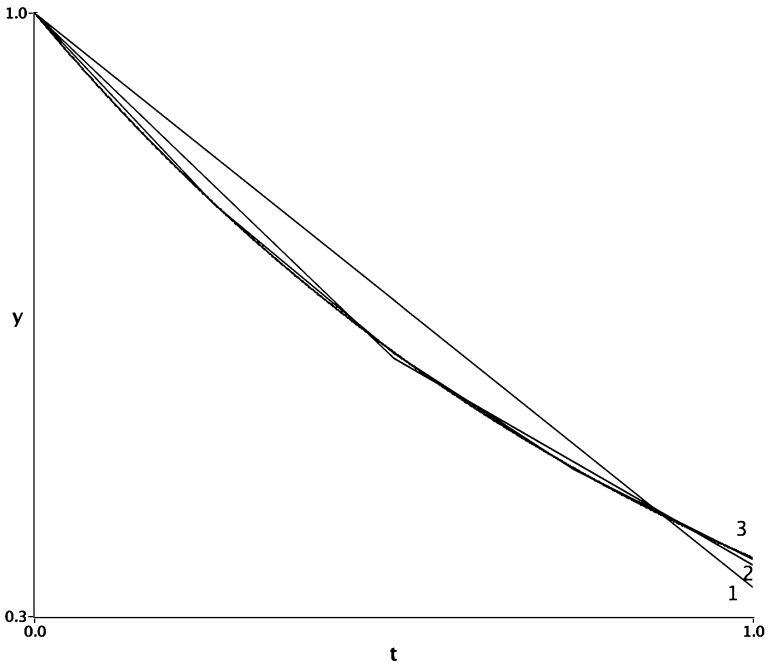


Figure 5.10. Behavior of the trapezoidal method: Accuracy.

so $y_n = ((1+w/2)/(1-w/2))^n y_o$. Using a geometric series expansion, $(1-w/2)^{-1} = 1+(w/2)+(w/2)^2+(w/2)^3\cdots$, $|w/2| < 1$, so for small $w$, one step of the trapezoidal method applied to the model problem, $(1+w/2)(1-w/2)^{-1} = 1+(w/2)+(w/2)^2+\cdots+(w/2)+(w/2)^2+(w/2)^3\cdots = 1+w+w^2/2+w^3/4+\cdots$, captures the terms of order $\leq w^2$ in the exact solution $y_n e^w$, and the remainder is bounded by a multiple of $w^3$.

In Figure 5.10, the trapezoidal method is employed with the same parameters as in Figure 5.3. As usual, each time $h$ decreases by factors of 1/2, the number $N$ of steps doubles. At $t_N = T = 1$, as with the midpoint and leapfrog methods, the difference between the approximate solution and the analytical solution appears to decrease by a factor of 1/4, suggesting that the order of accuracy for the trapezoidal method is also 2. The approximations with $j \geq 4$, $h \leq 1/16$ are

indistinguishable from the exact solution. In the same fashion, Figure 5.11 corresponds to Figure 5.4. At increasingly negative values of $\lambda$, we begin to observe the numerical solution becoming oscillatory and decaying less rapidly, even though the analytical solution continues to decay monotonically and more rapidly. This should be expected, since the factor $(1 + w/2)(1 - w/2)^{-1} \to -1$ as $w \to -\infty$. From the form of the solution above, we should expect absolute stability when $|(1 + w/2)(1 - w/2)^{-1}| \leq 1$, and instability otherwise. Rewriting this condition as $|w - (-2)| < |w - 2|$, we see that the trapezoidal method should be absolutely stable for any value of $w$ that is closer to $-2$ than to $+2$. The trapezoidal method is absolutely stable with respect to the model problem for any $w \leq 0$. Below, when we consider the complex scalar model problem, equivalent to the real $2 \times 2$ model problem in the plane, we will see that the region of absolute stability for the trapezoidal method is the closed left half-plane $\{w \in \mathbf{C} \mid \mathrm{Re}(w) \leq 0\}$.

A method that is absolutely stable for any complex $w$ whose real part is negative is known as an *A-stable* method.

• **Example 5–5. The modified trapezoidal method (aka Heun's Method and the improved Euler Method).** We can approximate the solution of the nonlinear equation that defines the trapezoidal method (5.22) quite easily if we approximate the value of $y_{n+1}$ on its right-hand side by using one step of Euler's Method and solve for $y_{n+1}$ as follows:

$$\begin{aligned} \bar{y}_{n+1} &= y_n + hf(t_n, y_n), \\ y_{n+1} &= y_n + h\frac{f(t_n, y_n) + f(t_{n+1}, \bar{y}_{n+1})}{2}, \quad n = 0, 1, \ldots \end{aligned} \tag{5.23}$$

This is another example of the explicit Runge-Kutta Methods and is known by many names, including the modified trapezoidal method, Heun's Method, and the improved Euler Method. Heun's Method is an explicit $r$-stage Runge-Kutta Method, with $r = 2, \gamma_1 = \gamma_2 = 1/2$, $\beta_{11} = \beta_{12} = 0, \beta_{21} = 1$, and $\beta_{22} = 0$. When Heun's Method is applied to the stability model problem $(M_S(\lambda))$, it coincides with the midpoint method, and so the results of the exercise will be identical and their regions of absolute stability are the same. When Heun's Method is applied to the accuracy model problem (5.19), it coincides

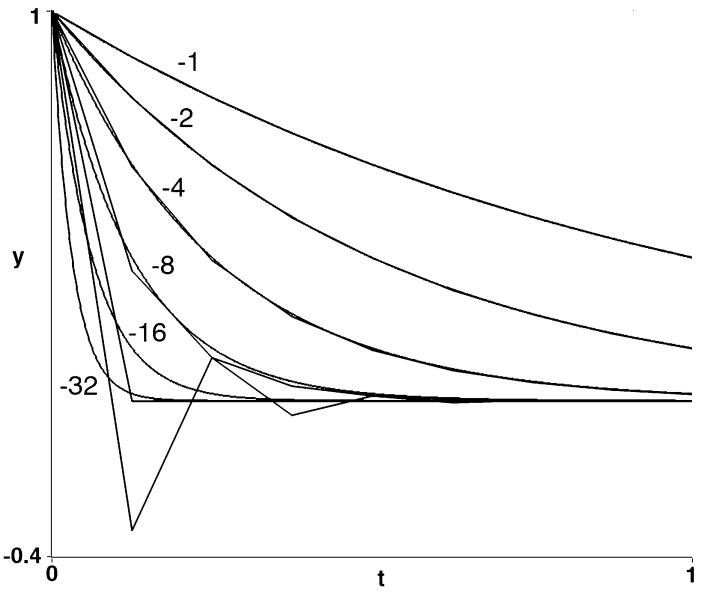


Figure 5.11. Behavior of the trapezoidal method: Stability.

with the trapezoidal method and so its formal order of accuracy is also 2.

We can also view (5.23) as a *predictor-corrector* method associated with the trapezoidal method (5.22). It may be worthwhile to solve the nonlinear equations associated with an implicit method using higher-order Newton-Raphson and quasi-Newton algorithms. These will usually require problem-specific implementations for evaluating or approximating and inverting derivatives involving considerable overhead. For universality and simplicity it is often preferable to take advantage of the natural fixed-point form

$$y_{n+1} = g(y_{n+1}; h, y_n, \ldots, y_{n+1-m})$$

of implicit numerical methods. To solve this using fixed-point iteration, we apply an explicit method called the 'predictor' to initialize $y_{n+1}^{(0)}$. For example,

$$y_{n+1}^{0} = y_n + hf(t_n, y_n) \tag{5.24}$$

is called an Euler predictor step. Then we apply one or more 'corrector' steps, i.e., steps of the associated iteration algorithm

$$y_{n+1}^{(k+1)} = g(y_{n+1}^{(k)}; h, y_n, \ldots, y_{n+1-m}).$$

For (5.23), each step of

$$y_{n+1}^{(k+1)} = g(y_{n+1}^{(k)}) = y_n + h\frac{f(t_n, y_n) + f(t_{n+1}, y_{n+1}^{(k)})}{2} \tag{5.25}$$

is called a trapezoidal 'corrector' step. When (5.24) is followed by one trapezoidal corrector step, the result is (5.23). So one more name for Heun's Method is the *Euler predictor-trapezoidal corrector* method.

The Lipschitz constant of the fixed-point iteration mapping approaches the magnitude of the derivative of the iteration function at the fixed point, $|g'(y_{n+1})|$. The iterations obtained from implicit numerical methods depend on the parameter $h$ and have $y_n$ as a fixed point for $h = 0$. Since $|g'(y_{n+1})|$ contains a factor of $h$, $g$ will be a contraction on some neighborhood of $y_n$ provided $h$ is sufficiently small, and for any $y_{n+1}^{(0)}$ in this neighborhood, $y_{n+1}^{(k)} \to y_{n+1}$ as $k \to \infty$. If we denote the 'local solution' of the ODE $y' = f(t, y)$ passing through $(t_n, y_n)$ by $\hat{y}_n(t)$, a $Q$th-order accurate predictor produces an initial approximation whose local error $|y_{n+1}^{(0)} - \hat{y}_n(t_{n+1})|$ has order of magnitude $h^{Q+1}$. A $P$th-order accurate predictor produces a converged approximation whose local error $|y_{n+1}^{(\infty)} - \hat{y}_n(t_{n+1})|$ has order of magnitude $h^{P+1}$. By the triangle inequality, we can estimate $|y_{n+1}^{(\infty)} - y_{n+1}^{(0)}| \leq Ch^{\min\{P+1,Q+1\}}$. Additional factors of $h$ from the iteration toward $y_{n+1}^{(\infty)}$ only decrease the magnitude of the local error for $k \leq P - Q$. For example, the Euler predictor-trapezoidal corrector method attains the full accuracy that would be obtained by iterating the trapezoidal corrector to convergence. But in problems where absolute stability is crucial, the difference in performance is substantial.

The modified trapezoidal method is only absolutely stable if $h$ is sufficiently small, while the trapezoidal method is absolutely stable for any $h$. This additional labor that characterizes the implicit method makes no difference at all to the order of accuracy, but all

the difference in the world to absolute stability. The extra effort of each corrector iteration often pays itself back with interest by further relaxing the absolute stability restriction on $h$.

The local errors of a $P$th-order accurate linear multistep method have a specific asymptotic form $C\hat{y}_n^{(P+1)}(t_n)h^{P+1}$ that can be determined from the coefficients of the method. This makes it possible to use a predictor and corrector of the same order to estimate the error of both methods efficiently, provided their constants $C$ are distinct. For example, the leapfrog predictor coupled with one trapezoidal corrector iteration permits efficient error estimation and stabilization, even though it does not add even one degree of accuracy to the predictor.

In contrast, the difference in local errors of different Runge-Kutta Methods of the same order $P \geq 2$ will in general be different for different ODEs. This makes it difficult to use such a pair for error estimation, and methods of different order are used instead. The Euler predictor-trapezoidal corrector pair serves as a prototype of Runge-Kutta error estimation. The local error of an order $P-1$ predictor is $|y_{n+1}^{(0)} - \hat{y}_n(t_{n+1})| \approx Ch^P$. The local error of a corrector of order $P$ is $|y_{n+1}^{(1)} - \hat{y}_n(t_{n+1})| \approx Ch^{P+1}$. Therefore, the correction $|{y_{n+1}}^{(1)} - {y_{n+1}}^{(0)}|$ is an order $h^{P+1}$ accurate estimate of the local error of the *lower* order method. For an Euler predictor-trapezoidal corrector pair, this technique estimates the error of the Euler step, even though we advance using the corrected Heun step. The resulting estimate is conservative, as one would want it. This approach is also efficient, since the lower-order method is *embedded*; i.e., it only involves evaluations performed by the higher-order method.

• **Example 5–6. The Backward Euler Method.** Another example of an implicit method is the Backward Euler Method,

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}), \quad n = 0, 1, \ldots, \tag{5.26}$$

that arises by replacing the left endpoint approximation that characterizes Euler's Method with the right endpoint approximation. The Backward Euler Method is an implicit linear $m$-step method with $m = 1$, $a_0 = 1$, $b_{-1} = 1$, and $b_0 = 0$. It is also an explicit $r$-stage Runge-Kutta Method with $r = 1$, $\gamma_1 = 1$, and $\beta_{11} = 1$.

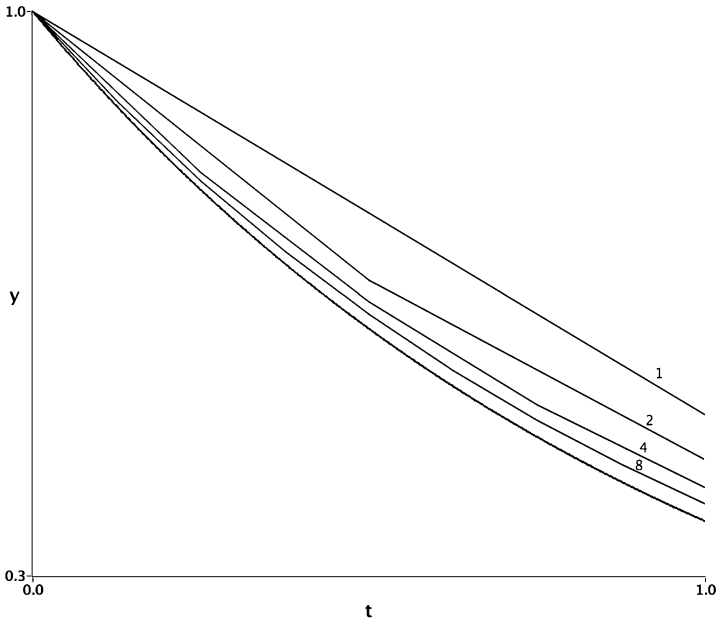


Figure 5.12. Behavior of the Backward Euler Method: Accuracy.

▷ **Exercise 5–7.** Modify the program implementing Euler's Method to implement the Backward Euler Method on the model problem $(M_S(\lambda))$, using the same parameters and using the options described above for implementing the trapezoidal method.

To understand the accuracy of the Backward Euler Method analytically, we use the same class of accuracy model problems $(M_A^2)$ in the same form (5.12), (5.12′) that we used to analyze Euler's Method. When the Backward Euler Method is applied to (5.12), it takes the form $y_{n+1} = y_n + h(c_1 + 2c_2(n+1)h)$. Using $\sum_{n=0}^{N-1} 2(n+1) = N^2 + N$, we find that $y_N = y_0 + c_1 Nh + c_2 h^2(N^2 + N)$, or in terms of $t_n - t_o$, $y_N = y_0 + c_1(t_n - t_o) + c_2(t_n - t_o)^2 + c_2 h(t_n - t_o)$. From this, we see that the global error at time $T = Nh$ satisfies $y(t_o + T) - y_N = (y_o - y_0) - c_2 Th$. The method is exact on polynomials of degree 1, and for a polynomial of degree 2, its error at a fixed $T$ is proportional to $h$. In the general case the bound involves a factor $\max_{t\in[t_o,t_o+T]} \frac{y''(t)}{2}$

that reduces to the factor of $c_2$ above. Note that the errors of Euler's Method and the Backward Euler Method have the same magnitude but opposite signs on these problems. We can show that the leading order errors in these methods are opposite in general and obtain a more accurate method by averaging them. In its general form, this process is known as extrapolation. Extrapolation by averaging Euler's Method and the Backward Euler Method is an alternate approach to deriving the trapezoidal method.

When the Backward Euler Method is applied to the absolute stability model problems ($M_S(\lambda)$), it takes the form

$$y_{n+1} = (1 - w)^{-1} y_n$$

whose solution is $y_n = (1/(1 - w))^n y_o$. Using a geometric series expansion, $(1 - w)^{-1} = 1 + w + w^2 + \cdots$, $|w| < 1$, so for small $w$, the Backward Euler Method captures the terms of order $\leq w^1$ in the exact solution $y_n e^w$, and the remainder is bounded by a multiple of $w^2$.

In Figure 5.12, the Backward Euler Method is employed with the same parameters as in Figure 5.3, as $h$ decreases by factors of $1/2$, the number of steps doubles. As with Euler's Method, at $t_N = T = 1$, the difference between the approximate solution and the analytical solution appears to decrease by a factor of $1/2$, suggesting that the Backward Euler Method has the same first-order accuracy as Euler's Method.

In the same fashion Figure 5.13 corresponds to Figure 5.4. Unlike the trapezoidal method, at increasingly negative values of $\lambda$, the Backward Euler Method does not generate oscillations. The factor $(1 - w)^{-1} \to 0^+$ as $w \to -\infty$. From the form of the solution above, we should expect absolute stability when $|(1/(1 - w))| \leq 1$ and instability otherwise. Rewriting this, the region of absolute stability of the Backward Euler Method is $\{w \in \mathbf{C} \mid |w - 1| \geq 1\}$; i.e., the Backward Euler Method should be absolutely stable for any complex value of $w$ outside the circle of radius 1 centered at $w = 1$. Like the trapezoidal method, the Backward Euler Method is A-stable. Note that in the right half-plane outside this circle, the Backward Euler

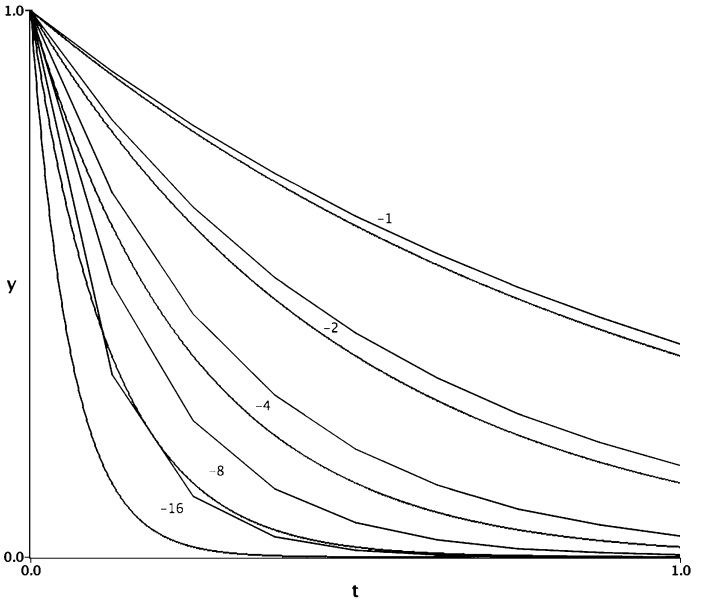


Figure 5.13. Behavior of the Backward Euler Method: Stability.

Method has decaying solutions where the analytic solution is growing exponentially, the opposite problem to what we observed previously!

Implicit linear $m$-step methods with $m > 1$ can be obtained using elementary means by simply doubling the discretization parameter of the trapezoidal and Backward Euler Methods:

$$y_{n+1} = y_{n-1} + 2h\frac{f(t_{n-1}, y_{n-1}) + f(t_{n+1}, y_{n+1})}{2}, \quad n = 0, 1, \ldots . \tag{5.27}$$

$$y_{n+1} = y_{n-1} + 2hf(t_{n+1}, y_{n+1}), \quad n = 0, 1, \ldots . \tag{5.28}$$

We refer to (5.27) as the $2h$ trapezoidal method and to (5.28) as the $2h$ Backward Euler Method. We do not have to repeat the accuracy studies for these methods since the results are easily obtained from and are essentially the same as that of their 1-step relatives. Stability is a different story. The approximations $y_n$ with even and odd $n$ are completely independent, and when these methods are applied to the

model problem, both now possess an additional mode of the form $-1 + O(\lambda h)$. Continuing this approach to $mh$ methods only make things worse. Any method of the form $y_{n+1} = y_{n+1-m} + h(\cdots)$ will have $m$ independent modes of the form $e^{2\pi ij/m} + O(h)$ forming a basis of solutions when it is applied to the model problem, resulting in additional instability. Less contrived and better behaved examples of implicit (and explicit) 2-step methods are derived and analyzed in Appendix I.

• **Example 5–7. The $y$-midpoint method (also known as the implicit midpoint method).** Our final example is another implicit method that has features of midpoint and trapezoidal methods. We will call it the $y$-midpoint method:

$$y_{n+1} = y_n + hf(t_n + \frac{h}{2}, \frac{y_n + y_{n+1}}{2}), \quad n = 0, 1, \ldots . \tag{5.29}$$

We may also write this method in the form

$$y_{n+1} = y_n + hy'_{n,1}, \text{ where } y'_{n,1} = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}y'_{n,1}) \tag{5.29$'$}$$

that exhibits it as an $r$-stage Runge-Kutta Method, with $r = 1$, $\gamma_1 = 1$, and $\beta_{11} = 1/2$. The form (5.29) shows that when the $y$-midpoint method is applied to the stability model problem ($M_S(\lambda)$), it coincides with the trapezoidal method and so the results of the exercise will be identical and their regions of absolute stability the same. When the $y$-midpoint method is applied to the accuracy model problem (5.19), it coincides with the ordinary midpoint method and so its formal order of accuracy is also 2.

## 5.3. Summary of Method Behavior on Model Problems

To review what we have seen, all seven example methods satisfy the condition of 0-stability, because $w = 0$ is in their regions of absolute stability. Two of the seven, Euler's Method and the Backward Euler Method have formal accuracy of order 1, and the remaining five have formal accuracy of order 2. Based on just these facts, rigorous theory will show that they are all convergent and either first- or second-order accurate, respectively. We summarize the formulas for one step of each of our example methods, along with their order of accuracy

and the expressions for amplification factors that we have obtained from analytical study of the absolute stability model problem in Table 5.1.

The regions of absolute stability obtained by bounding the magnitude of the amplification factors by 1 are depicted in the shaded regions of Figure 5.14. (The form of the regions for the leapfrog method and higher-order Runge-Kutta Methods depend on analyses given below.)

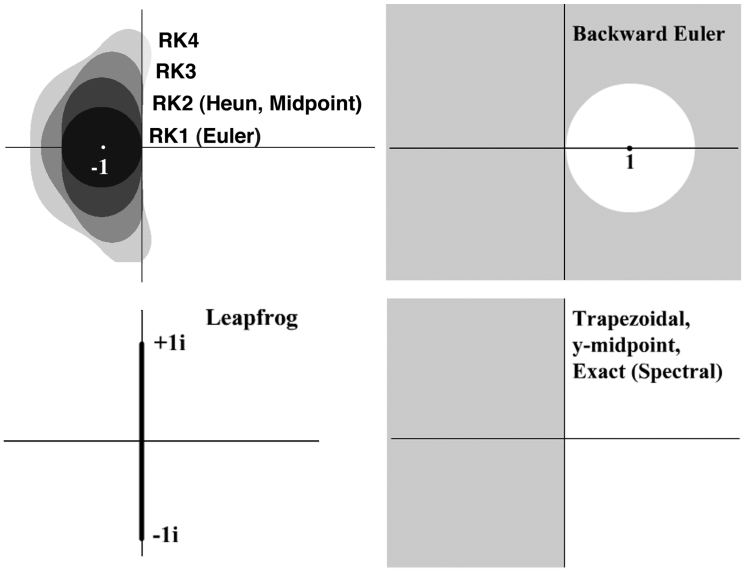


Figure 5.14. Regions of absolute stability for example methods.

The fact that $w = 0$ is in the region of absolute stability for six of the methods can be viewed as a simple consequence of the fact that they are one-step methods (and also Runge-Kutta Methods). For $y' = 0$ or just $h = 0$, they reduce to $y_{n+1} = y_n$. For a multistep method such as the leapfrog method, more analysis was required. In the introduction, we referred to a condition associated with the multistep method (5.4) when it is applied to the absolute stability model problem ($M_S(\lambda)$). In this situation, (5.4) reduces to a linear,

Table 5.1

| Method (order $p$) | $y_{n+1} =$ ($m$-steps, $r$-stages) | $a(w)$, $w = h\lambda$ |
|---|---|---|
| Euler (1) | $y_n + hf(t_n, y_n)$ $(1,1)$ | $1 + w$ |
| Backward Euler*(1) | $y_n + hf(t_{n+1}, y_{n+1})$ $(1,1)$ | $(1-w)^{-1}$ |
| Midpoint (2) | $y_n + hf(t_n + h/2, y_n + hf(t_n, y_n)/2)$ $(1,2)$ | $1 + w + w^2/2$ |
| Leapfrog (2) | $y_{n-1} + 2hf(t_n, y_n)$ $(2,1)$ | $w \pm \sqrt{1+w^2}$ |
| Trapezoidal*(2) | $y_n + h(f(t_n, y_n) + f(t_{n+1}, y_{n+1}))/2$ $(1,2)$ | $(1 + w/2)(1 - w/2)^{-1}$ |
| Heun (2) | $y_n + h(f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n)))/2$ $(1,2)$ | $1 + w + w^2/2$ |
| $y$-Midpoint*(2) | $y_n + hf(t_n + h/2, (y_n + y_{n+1})/2)$ $(1,2)$ | $(1 + w/2)(1 - w/2)^{-1}$ |

*Implicit method