Lecture Notes

Numerical Solution of ODEs

Davoud Mirzaei

Uppsala University

February 16, 2023

Contents

Welcome to a *beautiful* subject in scientific computing: numerical solution of ordinary differential equations (ODEs) with initial conditions. More precisely, we consider

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0, \quad t \in [t_0, b]$$

where t is an independent variable (usually plays the role of time), y is the unknown solution of the problem (possibly a vector) which is sought, f is a known function and y_0 is the initial condition at $t = t_0$. We first give a brief introduction to the theory of ODEs and existence and uniqueness of solutions. Then some standard numerical techniques are derived and their advantages and disadvantages for solving different differential equations are outlined. In some parts of this lecture we follow [**Atkinson-et-al:2009**] and [**LeVeque:2007**].

1 An introduction to ODEs

The simplest ordinary differential equation (ODE) has the form

$$y'(t) = g(t) \tag{1.1}$$

for a given function g. The *general solution* of this equations is

$$y(t) = \int g(\tau) d\tau + c$$

with c an arbitrary integration constant. Here $\int f(\tau)d\tau$ denotes any fixed antiderivative of g. In Figure 1, with the case of $g(t) = \cos 5t$, the plots of $y(t)$ for four different values of c are shown. Such plots are sometimes called *integration curves*. For the simple ODE (1.1), the integration curves are just copies (shifts) of each other along the y-axis.

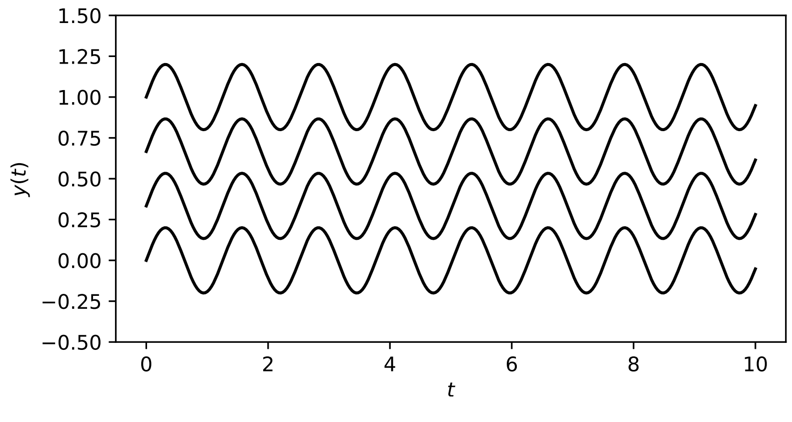


Figure 1: Integration curves

The constant c can be obtained by specifying the value of y at some given point

$$y(t_0) = y_0,$$

where y_0 is known. The *particular solution* of the differential equation then can be written as

$$y(t) = y_0 + \int_{t_0}^{t} g(\tau) d\tau,$$

provided that g be (for example) a continuous function. Another simple differential equation is

$$y'(t) = \lambda y(t), \quad y(t_0) = y_0, \quad \lambda \in \mathbb{R}$$

which possesses the exponential solution

$$y(t) = y_0 e^{\lambda(t-t_0)}.$$

A more general form which contains both the above differential equations reads as

$$y'(t) = \lambda y(t) + g(t), \quad y(t_0) = y_0. \tag{1.2}$$

The general solution of this equation can be obtained by the so-called *method of integration factors.* Multiplying the above linear equation by integration factor $e^{-\lambda t}$, the equation is reformulated as

$$\frac{d}{dt}\Big(e^{-\lambda t}y(t)\Big) = e^{-\lambda t}g(t).$$

Integrating both sides from t_0 to t, we obtain

$$y(t) = e^{\lambda t}\left[c + \int_{t_0}^{t} e^{-\lambda\tau}g(\tau)d\tau\right].$$

Imposing the condition $y(t_0) = y_0$ gives $c = \exp(-\lambda t_0)y_0$, and therefore

$$y(t) = y_0 e^{\lambda(t-t_0)} + \int_{t_0}^{t} e^{\lambda(t-\tau)}g(\tau)d\tau. \tag{1.3}$$

In many applications of differential equations the independent variable t plays the role of time, and t_0 can be interpreted as the initial time, and $y(t_0) = y_0$ is referred to the *initial condition.* A differential equation of the above type is called an **initial value problem (IVP)**. In a more general form an IVP has the form

$$\begin{aligned} \boldsymbol{y}'(t) &= \boldsymbol{f}(t, \boldsymbol{y}(t)) \\ \boldsymbol{y}(t_0) &= \boldsymbol{y}_0 \end{aligned} \tag{1.4}$$

which is a system of IVPs for vector $\boldsymbol{y} = [y_1, \ldots, y_n]^T$. Here $\boldsymbol{f}$ may represent a nonlinear relation between the independent variable t and the dependent variable $\boldsymbol{y}$.

For the linear IVP (1.2) the solution is given by (1.3) and it exists and is unique on any open interval where the data function f is continuous. But for the nonlinear IVP (1.4) even if the right hand side function $f(t, y)$ has derivatives of any order the solution of IVP may exist on only a smaller interval. In some cases the solution is not unique.

Example 1.1. Consider the nonlinear equation

$$y'(t) = -y(t)^2, \quad t \geqslant 0.$$

This problem has the trivial solution $y(t) \equiv 0$ and a general solution

$$y(t) = \frac{1}{t+c}$$

with arbitrary constant c. Let the equation be accompanied by initial condition $y(0) = y_0$. If $y_0 = 0$ then $y(t) \equiv 0$ is the solution of the IVP for any $t \geqslant 0$. If $y_0 \neq 0$ then the solution of IVP is

$$y(t) = \frac{1}{t + y_0^{-1}}.$$

For $y_0 > 0$ the solution exists for any $t \geqslant 0$ while for $y_0 < 0$ the solution exists only on interval $[0, -y_0^{-1}]$.

Example 1.2. Consider the IVP

$$y'(t) = 2\sqrt{y(t)}, \quad t \geqslant 0, \quad y(0) = 0.$$

It is clear that both $y(t) \equiv 0$ and $y(t) = t^2$ are solutions of this IVP. In additions any C^2 function $y(\cdot; \alpha)$ of the form

$$y(t;\alpha) = (t-\alpha)_+^2 = \begin{cases} 0, & 0 \leqslant t \leqslant \alpha \\ (t-\alpha)^2, & t > \alpha \end{cases}$$

for any $\alpha \geqslant 0$ is a solution for this IVP. In Figure 2 the solution with $\alpha = 0, 1, 2$ are plotted. This example reveals the non-uniqueness of the nonlinear IVP (1.4) for some right hand side functions f.

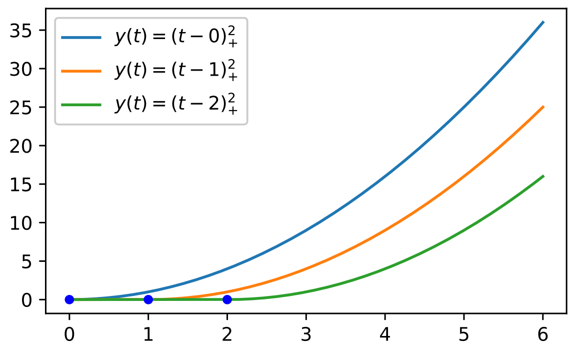


Figure 2: Three sample solutions for IVP $y' = 2\sqrt{y}$ with $y(0) = 0$ (non-uniqueness).

To guarantee that there is a unique solution it is necessary to impose a certain amount of smoothness on function $f(t, y)$. The following well-known theorem establishes the existence and uniqueness of the IVP (1.4).

Theorem 1.1. Let D be an open and connected set in $\mathbb{R}^2$ and $f(t, y)$ be a continuous function in both t and y in D, and let (t_0, y_0) be an interior point in D. Assume that f satisfies the Lipschitz continuity in its second argument, i.e., there exists a constant $L \geqslant 0$ such that

$$|f(t,y) - f(t,\tilde{y})| \leqslant L|y - \tilde{y}|, \quad \forall (t,y),\, (t,\tilde{y}) \in D.$$

Then there exists a unique function $y(t)$ defined on an interval $[t_0 - \beta, t_0 + \beta]$ for some $\beta > 0$ satisfying

$$y'(t) = f(t, y(t)), \quad t_0 - \beta \leqslant t \leqslant t_0 + \beta, \quad y(t_0) = y_0.$$

The Lipschitz continuity is slightly stronger than mere continuity, which only requires that

$|f(t,y) - f(t,\tilde{y})| \to 0$ as $\tilde{y} \to y$. Lipschitz continuity requires that

$$|f(t,y) - f(t,\tilde{y})| = \mathcal{O}(|y - \tilde{y}|), \; as \; \tilde{y} \to y.$$

If f is differentiable with respect to y in D and this derivative $\frac{\partial f}{\partial y}(t,y)$ is bounded then we can take

$$L = \max_{(t,y)\in\overline{D}} \left| \frac{\partial f}{\partial y}(t,y) \right|$$

because the Taylor series representation gives

$$f(y,t) = f(t,\tilde{y}) + (y - \tilde{y})\frac{\partial f}{\partial y}(t,\eta), \text{ for some } (t,\eta) \in D.$$

The number β in the statement of the theorem depends on the IVP (1.4). For some equations, solutions exist for any t, thus we can take β to be infinity. However for many nonlinear equations solutions can exist only in a bounded interval.

Example 1.3. For IVP (1.2) we have $f(t,y) = \lambda y + g(t)$; hence $L = |\lambda|$. This problem of course has a unique solution for any initial y_0 and for any t. In particular, if $\lambda = 0$ then $f(t,y) = g(t)$. In this case f is independent of y. The solution is then obtained by simply integrating the function $g(t)$.

Example 1.4. Consider the IVP

$$y'(t) = 2ty(t)^2, \quad y(0) = 1.$$

For this equation $f(t,y) = 2ty^2$ and $\frac{\partial f}{\partial y}(t,y) = 4ty$. Both of this functions are continuous for all (t,y). On any bounded domain $D = (a,b) \times (c,d)$ containing $(t_0, y_0) = (0,1)$ we can take $L = 4bd$. According to Theorem 1.1 there is a unique solution to this IVP for t in some neighborhood of $t_0 = 0$. This solution is

$$y(t) = \frac{1}{1-t^2}, \quad -1 < t < 1.$$

As a side note, this example shows that the continuity of $f(t,y)$ and $\frac{\partial f}{\partial y}(t,y)$ for all (t,y) does not imply the existence of a solution $y(t)$ for all t.

1.1 Modelling with ODEs: a funny example

Imagine that you are jogging along a given path. Suddenly a dog in a nearby garden sees you and begins chasing you at full speed with constant velocity w. What is the trajectory of the dog if we assume it is always running directly toward you?[1]

This situation is depicted in Figure 3. We assume that the trajectory of you is represented by $(\xi(t), \eta(t))$ and the trajectory of dog by $(x(t), y(t))$. Since the dog is running with constant speed w, we have

$$[x'(t)]^2 + [y'(t)]^2 = w^2, \quad \forall t \geqslant 0. \tag{1.5}$$

[1]This example is taken form: W. Gander, M. J. Gander, F. Kwok, Scientific Computing, An Introduction using Maple and MATLAB, Springer (2014).

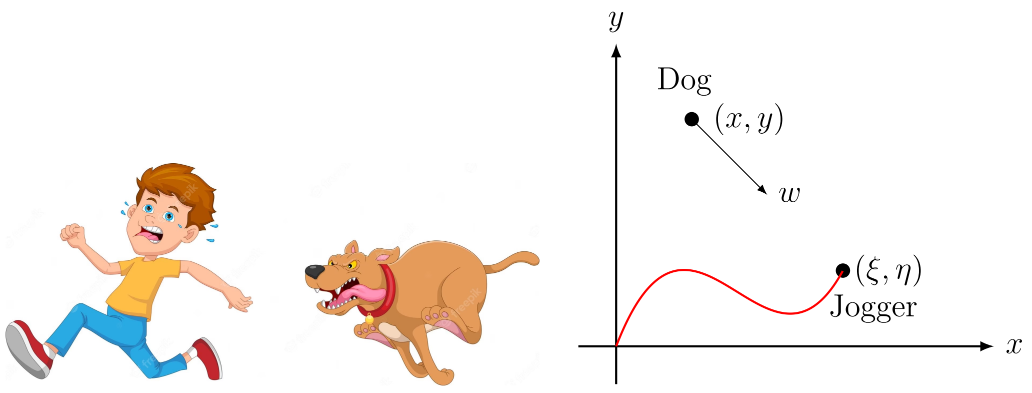


Figure 3: Dog chasing a jogger. (left image from freepik.com)

Since the dog is always running toward you, the velocity vector of the dog is proportional to the difference vector between the position of you and the dog, i.e.,

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \lambda(t) \begin{bmatrix} \xi(t) - x(t) \\ \eta(t) - y(t) \end{bmatrix}, \quad \forall t \geqslant 0, \tag{1.6}$$

where $\lambda(t) > 0$ is a constant of proportionality. To find $\lambda(t)$ we can substitute (1.6) in to (1.5) to obtain

$$\lambda^2 = \frac{w^2}{(\xi - x)^2 + (\eta - y)^2}.$$

The trajectory of the dog therefore satisfies the following system of ODEs

$$\begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix} = \frac{w}{\sqrt{(\xi(t) - x(t))^2 + (\eta(t) - y(t))^2}} \begin{bmatrix} \xi(t) - x(t) \\ \eta(t) - y(t). \end{bmatrix} \tag{1.7}$$

The initial condition for this ODE is the initial position of the dog, i.e., $(x(0), y(0)) = (x_0, y_0)$.

To find the trajectory of the dog, it remains to solve this IVP. But there is no hope for finding a closed-form solution for a general jogging path $(\xi(t), \eta(t))$. Soon we will solve this equation using some numerical methods, but let see the dog trajectories for different choices of jogger's path in Figure 4. This results are obtained by the Euler's method. See section 2 below.

1.2 Higher order ODEs

The highest-order derivative appearing in an ODE determines the order of the ODE. A k-th order ODE in the most general form can be written as

$$f(t, y, y', \dots, y^{(k)}) = 0,$$

where $f : \mathbb{R}^{k+2} \to \mathbb{R}$ is a known function and $y(t)$ is to be determined. A k-th order ODE is said to be explicit if it can be written in the form

$$y^{(k)}(t) = f(t, y, y', \dots, y^{(k-1)}). \tag{1.8}$$

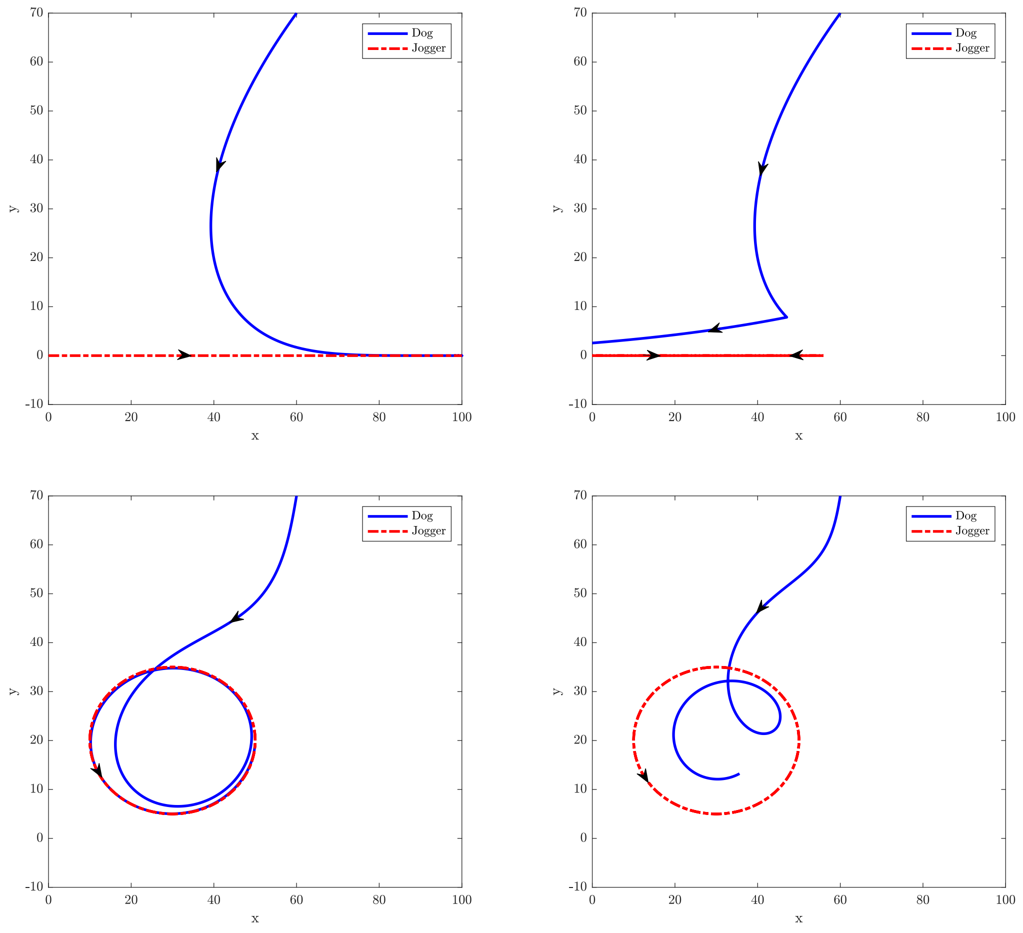


Figure 4: Jogging path and the numerically computed trajectory of the dog: The jogger running in a straight path (top-left), the jogger notices the dog and tries to run back (top-right), the jogger running on a circular track (bottom-left), the jogger running on a circular track but the dog is slow (bottom-right).

As a necessary condition for a unique solution, this ODE should accompany with k initial conditions

$$y(t_0) = y_0, \quad y'(t_0) = y_0', \quad \dots \quad y^{(k-1)}(t_0) = y_0^{(k-1)}. \tag{1.9}$$

Many ODEs arise naturally of the form (1.8), and many others can be transformed into it. So far we considered the case $k = 1$. This is not a real restriction because a higher-order ODE can always be reduced to a system of first-order equations as follows. For the explicit k-th order ODE (1.8) define k new variables

$$y_1(t) = y(t), \quad y_2(t) = y'(t), \quad \dots \quad y_k(t) = y^{(k-1)}(t),$$

so that the original k-th order equation becomes a system of k first-order equations

$$\boldsymbol{y}'(t) = \begin{bmatrix} y_1'(t) \\ y_2'(t) \\ \vdots \\ y_{k-1}'(t) \\ y_k'(t) \end{bmatrix} = \begin{bmatrix} y_2(t) \\ y_3(t) \\ \vdots \\ y_k(t) \\ f(t, y_1, y_2, \dots, y_k) \end{bmatrix} =: \boldsymbol{g}(t, \boldsymbol{y}),$$

with initial condition

$$y_1(t_0) = y_0, \quad y_2(t_0) = y_0', \quad \ldots \quad y_k(t_0) = y_0^{(k-1)}.$$

or simply

$$\boldsymbol{y}(t_0) = \boldsymbol{y}_0,$$

for $\boldsymbol{y}_0 = [y_0, y_0', \ldots, y_0^{(k-1)}]$. In general, we will focus on explicit first-order system of ODEs with initial conditions of the form

$$\begin{aligned} \boldsymbol{y}'(t) &= \boldsymbol{f}(t, \boldsymbol{y}) \\ \boldsymbol{y}(t_0) &= \boldsymbol{y}_0 \end{aligned} \tag{1.10}$$

where $\boldsymbol{f} : \mathbb{R}^{n+1} \to \mathbb{R}^n$ and $\boldsymbol{y}_0 \in \mathbb{R}^n$. If $\boldsymbol{f}$ is not explicitly depend on t, i.e., $\boldsymbol{f}(t, \boldsymbol{y}) = \boldsymbol{f}(\boldsymbol{y})$, the system is called *autonomous* and can be written in the form

$$\boldsymbol{y}' = \boldsymbol{f}(y).$$

A nonautonomous ODE $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$ can always be converted to autonomous form by introducing an additional dependent variable $y_{n+1}(t) = t$, so that $y_{n+1}'(t) = 1$ and $y_{n+1}(t_0) = t_0$ yielding the autonomous ODE

$$\begin{bmatrix} \boldsymbol{y}'(t) \\ y_{n+1}'(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}(y_{n+1}, \boldsymbol{y}) \\ 1 \end{bmatrix}.$$

It is often convenient to assume $\boldsymbol{f}$ is of this form since it simplifies notation.

Example 1.5. Consider the second order ODE

$$\theta''(t) = -\frac{g}{\ell} \sin(\theta(t)),$$

which models the motion of a pendulum with mass m at the end of a rigid bar of length ℓ by ignoring the mass of the bar and forces of friction and air resistance. Here $\theta(t)$ is the angle of the pendulum from vertical at time t, and g is the gravitational constant. The motion is independent of the mass of pendulum. Let $v = \theta'$ be the velocity and define

$$\boldsymbol{y} = \begin{bmatrix} \theta \\ v \end{bmatrix}$$

to obtain the following first-order linear system of equations

$$\begin{bmatrix} \theta' \\ v' \end{bmatrix} = \begin{bmatrix} v \\ -(g/\ell)\sin\theta \end{bmatrix} =: \begin{bmatrix} f_1(\theta, v) \\ f_2(\theta, v) \end{bmatrix}.$$

Workout 1.2. Convert the following system of third order equations to a system of first order equations:

$$\begin{aligned} &u'''(t) + 4u''(t) + 5u'(t) + 2u(t) = 2t^2 + 10t + 8 \\ &u(0) = 1, \ u'(0) = -1, \ u''(0) = 3. \end{aligned}$$

Workout 1.3. The following system of second order equations arises from studying the gravitational attraction of one mass by another. Convert it to a system of first order equations.

$$x''(t) = -\frac{cx(t)}{r(t)^3}, \quad y''(t) = -\frac{cy(t)}{r(t)^3}, \quad z''(t) = -\frac{cz(t)}{r(t)^3},$$

Here c is a positive constant and $r(t) = \sqrt{x(t)^2 + y(t)^2 + z(t)^2}$ with t denoting time.

1.3 First order system of ODEs

The study of first order system of ODEs is essentially important not only in solving a high order scaler equation using the techniques for first order ODEs but also in variety of other applications in which the system is obtained directly from the problem model. Such systems also appear in the procedure of solving parabolic and hyperbolic partial differential equations using the method of lines (MOL).

The system of ODEs (1.10) is linear if

$$\boldsymbol{f}(t, \boldsymbol{y}) = A(t)\boldsymbol{y} + \boldsymbol{g}(t)$$

where $A(t) \in \mathbb{R}^{n\times n}$ and $\boldsymbol{g} \in \mathbb{R}^n$. An important special case is the constant coefficient linear system

$$\boldsymbol{y}'(t) = A\boldsymbol{y}(t) + \boldsymbol{g}(t)$$

where $A \in \mathbb{R}^{n\times n}$ is a constant matrix. If $\boldsymbol{g}(t) = 0$, then the equation is homogeneous. The solution to the homogeneous system $\boldsymbol{y}'(t) = A\boldsymbol{y}(t)$ with initial data $\boldsymbol{y}(t_0) = \boldsymbol{y}_0$ is

$$\boldsymbol{y}(t) = e^{A(t-t_0)}\boldsymbol{y}_0.$$

where $e^{A(t-t_0)}$ is matrix exponential.

Example 1.6 (Chemical Reaction Kinetics)**.** Let X and Y represent chemical compounds and consider a reaction of the form

$$X \xrightarrow{k_1} Y$$

which represents a reaction in which X is transformed into Y with rate $k_1 > 0$. If we let y_1 represent the concentration of X and y_2 represent the concentration of Y (often denoted by $y_1 = [X]$ and $y_2 = [Y]$, then the ODEs for y_1 and y_2 are

$$\begin{aligned} y_1' &= -k_1 y_1 \\ y_2' &= +k_1 y_1 \end{aligned}$$

If there is also a reverse reaction at rate k_2, we write

$$X \underset{k_2}{\overset{k_1}{\rightleftarrows}} Y$$

and we have the system of ODEs

$$y_1' = -\,k_1 y_1 + k_2 y_2$$
$$y_2' = +\,k_1 y_1 - k_2 y_2$$

which with given initial concentrations $y_1(0)$ and $y_2(0)$ forms a linear system of initial value problems with constant coefficient matrix:

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} -k_1 & k_2 \\ k_1 & -k_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} y_{1,0} \\ y_{2,0} \end{bmatrix}. \tag{1.11}$$

Another simple system arises from the decay process

$$X \xrightarrow{k_1} Y \xrightarrow{k_2} Z.$$

If $y_1 = [X]$, $y_2 = [Y]$ and $y_3 = [Z]$ then we have the following equations

$$\begin{aligned} y_1' &= -k_1 y_1 \\ y_2' &= k_1 y_1 - k_2 y_2 \\ y_3' &= k_2 y_2 \end{aligned}, \quad \textit{or} \quad \begin{bmatrix} y_1' \\ y_2' \\ y_3' \end{bmatrix} = \begin{bmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}. \tag{1.12}$$

Consider a linear system $\boldsymbol{y}' = A\boldsymbol{y}$ with initial condition $\boldsymbol{y}(t_0) = \boldsymbol{y}_0$, where A is a constant $n \times n$ matrix, and suppose for simplicity that A is diagonalizable, i.e., A has a complete set of n linearly independent eigenvectors $\boldsymbol{v}_k$ corresponding to eigenvalues λ_k for $k = 1, \ldots, n$ such that

$$A\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k, \quad k = 1, \ldots, n$$

or equivalently

$$A = VDV^{-1},$$

where $V = [\boldsymbol{v}_1\ \boldsymbol{v}_2 \ldots \boldsymbol{v}_n]$ and $D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. Applying the change of variables $\boldsymbol{u}(t) = V^{-1}\boldsymbol{y}(t)$, the linear system $\boldsymbol{y}' = A\boldsymbol{y}$ will transfer to

$$\boldsymbol{u}' = D\boldsymbol{u}, \quad \boldsymbol{u}(t_0) = V^{-1}\boldsymbol{y}_0 =: \boldsymbol{u}_0.$$

This is a decoupled system of ODEs because D is diagonal. We may write

$$u_k' = \lambda_k u_k, \quad u_k(t_0) = u_{k,0}, \quad k = 1, 2, \ldots, n, \tag{1.13}$$

with solutions $u_k(t) = u_{k,0} e^{\lambda_k (t-t_0)}$ or in a matrix form

$$\boldsymbol{u}(t) = e^{D(t-t_0)} \boldsymbol{u}_0$$

or equivalently

$$V^{-1}\boldsymbol{y}(t) = e^{D(t-t_0)} V^{-1} \boldsymbol{y}_0$$

which finally gives

$$\boldsymbol{y}(t) = V e^{D(t-t_0)} V^{-1} \boldsymbol{y}_0 = e^{A(t-t_0)} \boldsymbol{y}_0.$$

Keep in mind that $e^{A(t-t_0)}$ is a matrix and $\boldsymbol{y}_0$ is a vector, thus $e^{A(t-t_0)}\boldsymbol{y}_0$ is a matrix-vector multiplication.

Example 1.7. Consider the linear system of ODEs (1.11) with $k_1 = 2$ and $k_2 = 1$ and initial conditions $y_1(0) = 5$ and $y_2(0) = 2$:

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}.$$

It can be simply shown that the eigenvalues and eigenvectors of A are

$$\lambda_1 = -3, \quad \boldsymbol{v}_1 = [1, -1]^T$$
$$\lambda_2 = 0, \quad \boldsymbol{v}_2 = [1, 2]^T,$$

which gives

$$V = \begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix}, \quad D = \begin{bmatrix} -3 & 0 \\ 0 & 0 \end{bmatrix}.$$

Therefore, the solution can be written as

$$\boldsymbol{y}(t) = V e^{Dt} V^{-1} \boldsymbol{y}_0 = \frac{1}{3} \begin{bmatrix} 1 & 1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} e^{-3t} & 0 \\ 0 & e^{0t} \end{bmatrix} \begin{bmatrix} 2 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

which shows that

$$\begin{aligned} y_1(t) &= \frac{5}{3}\left[2e^{-3t} + 1\right] + \frac{2}{3}\left[-e^{-3t} + 1\right] \\ y_2(t) &= \frac{5}{3}\left[-2e^{-3t} + 2\right] + \frac{2}{3}\left[e^{-3t} + 2\right]. \end{aligned}$$

The plots of solutions are depicted in Figure 5.

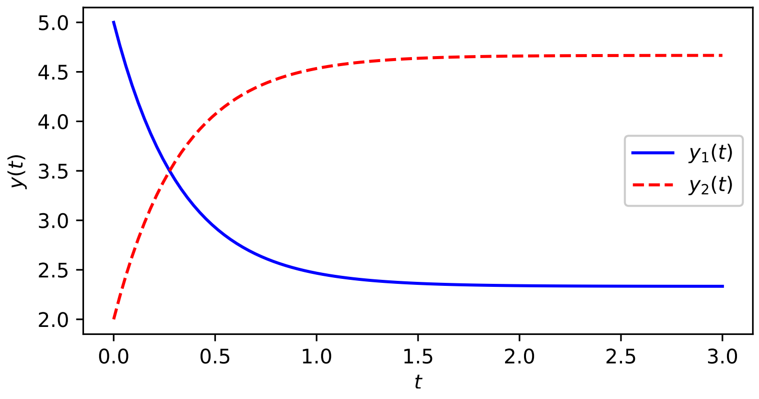


Figure 5: Solutions of a chemical reaction kinetics problem.

Since, in this example, the reaction rate k_1 is larger than k_2, the amount y_1 decreases while y_2 increases. Both solutions reach the steady states $y_1(t) \to \frac{1}{3}(y_1(0) + y_2(0)) = \frac{7}{3}$ and $y_2(t) \to \frac{2}{3}(y_1(0) + y_2(0)) = \frac{14}{3}$ when t increases.

Workout 1.4. In the chemical reaction model (1.12) let $k_1 = 2$ and $k_2 = 1$ and $[y_1(0), y_2(0), y_3(0)] = [1, 3, 2]$. Obtain the solution.
Hint: Calculate the eigenvalues and eigenvectors of the coefficient matrix and put them into the formula.

The situation will be more complicated for a nonlinear system of equations. In the sequel we study several numerical algorithms for solving different types of ODEs; both linear and nonlinear equations.

1.4 Stability of solutions

Depending on given data f and y_0, solutions of IVP may behave differently as $t \to \infty$. The Lipschitz constant measures how much $f(t, y)$ is changed if y is perturbed (at some fixed time t). Since $f(t, y) = y'(t)$ is the slope of the line tangent to the solution curve through the value y, this indicates how the slope of the solution curve is varied if y is perturbed.

Example 1.8. The solutions of $y'(t) = g(t)$ with Lipschitz constant $L = 0$ are parallel curves each for a prescribed initial condition y_0. See Figure 1. The ODE $y'(t) = \lambda y(t)$ with $L = |\lambda|$ possesses solutions $y(t) = y_0 e^{\lambda t}$. Depending on the sign of λ solutions decay exponentially to zero (if $\lambda < 0$), grow exponentially to infinity (if $\lambda > 0$), or stay parallel lines (if $\lambda = 0$) for different values of y_0. See Figure 6.

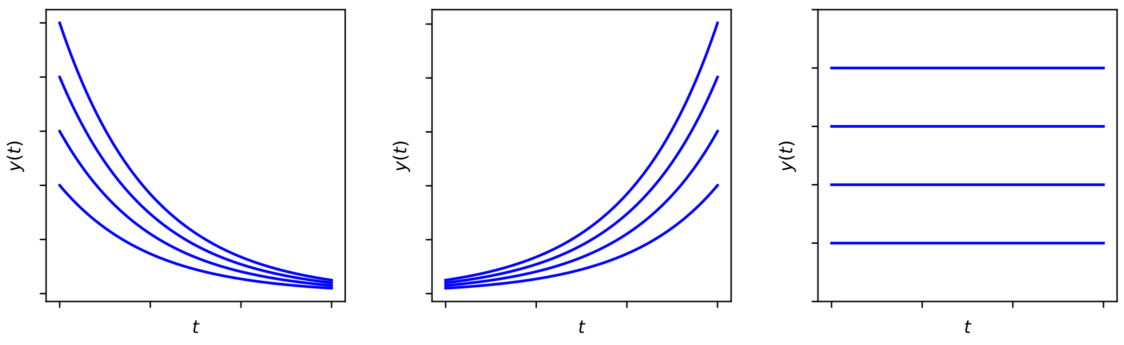


Figure 6: Solutions of IVP $y' = \lambda y$ with different initial values y_0 for $\lambda < 0$ (left), $\lambda > 0$ (middle), and $\lambda = 0$ (right).

For the case of $\lambda > 0$ any two solutions diverge away from each other, i.e., a small perturbation in the initial condition results in a substantial difference between the final solutions at time $t > t_0$. For the case of $\lambda < 0$ the situation is different; any two solutions converge toward each other and even large perturbation in initial data will finally diminish in the solutions. For the case of $\lambda = 0$ perturbations in data and solution are of the same order. If λ is a complex number, $\lambda = a + ib$ say, then

$$y(t) = y_0 e^{at}(\cos bt + i \sin bt).$$

The behaviour now depends on the sign of $\mathrm{Re}(\lambda) = a$. We have exponential decay for $\mathrm{Re}(\lambda) < 0$, exponential growth for $\mathrm{Re}(\lambda) > 0$ and oscillatory solutions (parallel curves) for $\mathrm{Re}(\lambda) = 0$.

A solution of the ODE $\boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y})$ is said to be **stable** if for every $\epsilon > 0$ there is a $\delta > 0$

such that if $\widehat{\boldsymbol{y}}(t)$ satisfies the ODE and $\|\widehat{\boldsymbol{y}}(t_0) - \boldsymbol{y}(t_0)\| \leqslant \delta$, then

$$\|\widehat{\boldsymbol{y}}(t) - \boldsymbol{y}(t)\| \leqslant \epsilon, \quad \text{for all} \quad t \geqslant t_0. \tag{1.14}$$

Hence, for a stable solution, if the initial value is perturbed, the perturbed solution remains close to the original solution. This roles out the exponential divergence solutions allowed by the ODE. A stable solution is said to be **asymptotically stable** if

$$\|\widehat{\boldsymbol{y}}(t) - \boldsymbol{y}(t)\| \to 0, \; as\; t \to \infty.$$

This stronger form of stability means that the original and perturbed solutions not only remain close to each other, they converge toward each other over time. As we will soon see in detail, the significance of these concepts for the numerical solution of ODEs is that any errors introduced during the computation can be either amplified or diminished over time, depending on the stability of the solution.

Workout 1.5. Consider the IVP

$$y'(t) = -[y(t)]^2, \quad y(0) = 1.$$

Show that the solution is $y(t) = 1/(1+t)$. Then solve the perturbed problem

$$\widehat{y}'(t) = -[\widehat{y}(t)]^2, \quad \widehat{y}(0) = 1 + \delta$$

and show this IVP is stable, and even asymptotically stable.
Hint: This is a separable differential equation, so the analytical solution can be easily obtained.

Example 1.9. For system of ODEs

$$\boldsymbol{y}' = A\boldsymbol{y}$$

for $n \times n$ constant diagonalizable matrix A, the stability of solutions depends on the sign of the real part of eigenvalues of A. See (1.13). In this case eigenvalues with negative real parts yield exponentially decaying solution components, eigenvalues with positive real parts yield exponentially growing solution components, and eigenvalues with zero real parts give oscillatory solutions. This means that the solutions of this ODE are stable if $\mathrm{Re}(\lambda_k) \leqslant 0$ for all $k = 1, 2, \ldots, n$, and asymptotically stable if $\mathrm{Re}(\lambda_k) < 0$ for all $k = 1, 2, \ldots, n$, but unstable if there is any eigenvalue such that $\mathrm{Re}(\lambda_k) > 0$.

For the general case $\boldsymbol{y}'(t) = A(t)\boldsymbol{y}(t)$ where $A(t)$ is a time dependent matrix or for the nonlinear equation $\boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y})$ the stability analysis is more complicated.

2 Basic numerical methods

Although the exact solution of some ODEs can be obtained by few analytic methods, such as method of integration factors, the solution of most ODEs arising in applications are so complicated that should be computed only by numerical methods. Even when a solution formula is available, it may involve integrals that can be calculated only by numerical integration formulas. An analytical solution of an ODE is a continuous (and sometimes a close form) function in an infinite-dimensional space, while a numerical solution is a table of approximate values of the solution function at a discrete set of points which can be considered as a vector in a finite dimensional space.

In this section some basic numerical methods are given for solving (1.2). In all methods, starting from the initial condition y_0 at $t = t_0$, the approximate solutions at times $t_1, t_2, \dots$ are obtained successively by solving an algebraic (system of) difference equations obtained by discretizing the differential equation.

2.1 Euler's method

The simplest technique is the **Euler's method**, also called *explicit* or *forward Euler's method*. For a given time step $h > 0$ assume that $t_k = t_0 + kh$, $k = 1, 2, \dots, N$, is a partitioning of time domain $[t_0, b]$ with $b = t_N$. Let the derivative $y'(t)$ in the ODE $y'(t) = f(t, y)$ at $t = t_k$ be approximated by the first-order forward difference approximation

$$y'(t_k) = \frac{y(t_{k+1}) - y(t_k)}{h} + \frac{h}{2} y''(\xi_k), \quad t_k \leqslant \xi_k \leqslant t_{k+1}.$$

By dropping the error term and using the approximate values y_k instead of $y(t_k)$, we obtain

$$\begin{aligned} &y_{k+1} = y_k + hf(t_k, y_k), \quad k = 0, 1, \dots, N-1, \\ &y_0 = y(t_0) \end{aligned} \tag{2.1}$$

We use $y_0 = y(t_0)$ or some close approximation of it. Formula (2.1) gives a rule for computing $y_1, y_2, \dots, y_N$ in succession. We bring an example from [**Heath:2018**].

Example 2.1. Consider the simple IVP $y' = y$ with initial value y_0 at initial time $t_0 = 0$. The approximate value y_1 is obtained as $y_1 = y_0 + hy_0 = (1 + h)y_0$. It is obvious that $y_1 \neq y(t_1)$, thus, the value y_1 lies on a different solution curve of the ODE from the one on which we started, as shown in the left side of Figure 7. The approximate value y_1 is the slope of the tangent line for the new (perturbed) solution curve. Now we continue to obtain the approximate value y_2 at $t = t_2$ by starting from y_1 with Euler rule $y_2 = y_1 + hy_1 = (1+h)y_1$. The approximate value y_2 carries both previous approximation error in y_1 and a new error introduced in the current step of the Euler's method. In fact, in this step we have moved to still another solution of the ODE, as is again shown in Figure 7. We can advance to future times $t_3, t_4, \dots$ until reaching the final time $b = t_N$. In each step a new *local truncation error* is introduced and the approximate solution falls down (or rises up) to another solution curve. Since the solutions of this ODE are unstable, the errors are amplified with time. For an equation with stable solutions, on the other hand, the errors in the numerical solution do not grow, and for an equation with asymptotically stable solutions, such as $y' = -y$, the errors diminish with time, as is shown in the right hand side of Figure 7.

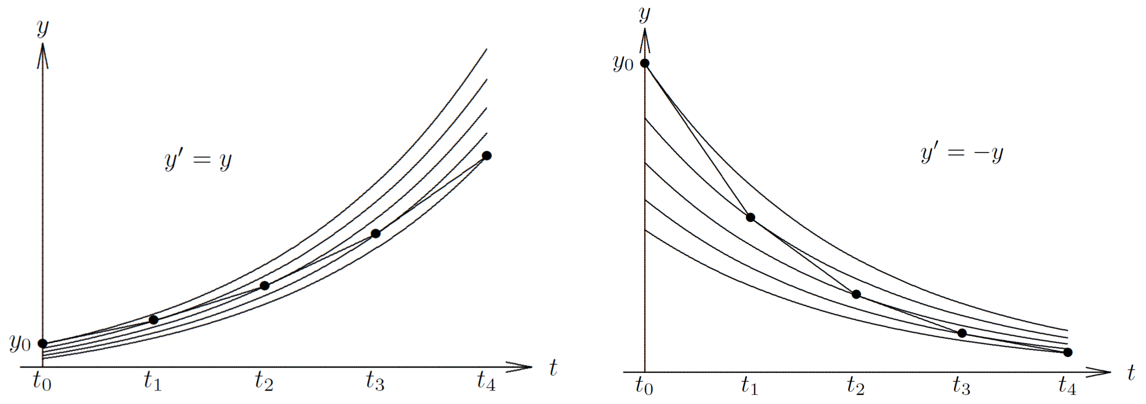


Figure 7: Steps of Euler's method for $y' = y$ (left) and $y' = -y$ (right).

Workout 2.1. Write down Euler's formula for the following IVPs. Compute 1 iteration for the third ODE with $h = 0.1$.

- (a) $y'(t) = te^{-t} - y(t),\ y(0) = 1,$
- (b) $y'(t) = [\cos(y(t))]^2,\ y(0) = 0,$
- (c) $y'(t) = t^3/y(t),\ y(0) = 1.$

If the ODE is a system then y and f in formula (2.1) are simply replaced by vectors $\boldsymbol{y}$ and $\boldsymbol{f}$. The ODE algorithms are usually simple to program. Here is the function of explicit Euler's method.

```matlab
function [T,Y] = ExEuler(f,y0,tspan,h)
% Euler's method for IVP system y' = f(t,y) with y(t0) = y0
% Inputs:
%  f: right hand side function f(t,y)
%  y0: initial condition of size (1 x m)
%  tspan: [t0, tfinal]
%  h: stepsize
% Output:
%  T: vector of time step
%  Y: solution of size (N x m)
Y = y0; T = tspan(1);
for t = tspan(1):h:tspan(2)-h
   y = y0 + h*f(t,y0);
   Y = [Y y]; y0 = y; T = [T t+h];
end
```

The input `f` is a function of t and y variables, and can be defined as a separated function in the main script. For example we use the following commands for solving the chemical reaction kinetics ODE (1.11) on interval $[0, 3]$ with $K_1 = 2$ and $K_2 = 1$ and initial conditions $y_1(0) = 5$ and $y_2(0) = 2$. We also set $h = 0.01$.

```matlab
%% Setup of the problem
y0 = [5;2]; tspan = [0 3]; h = 0.01;
% ODE call
[t,y] = ExEuler(@func,y0,tspan,h);
% plot solutions y1 and y2
plot(t,y(1,:),'-b',t,y(2,:),'--r')
set(gca,'TickLabelInterpreter','latex')
xlabel('Time $t$', Interpreter='latex');
ylabel('Solution $y$',Interpreter='latex');
leg = legend('$y_1$','$y_2$'); set(leg,Interpreter='latex');
%% definition of right-hand side function f
function yprime = func(t,y)
yprime = [-2*y(1) + y(2); 2*y(1)-y(2)];
end
```

The exact solution was given in Example 1.7 and Figure 5. If you run this script on your computer you will receive the same plot.

Lab Exercise 2.2. Solve the dog-jogger problem using the Euler's method and reproduce the plots of Figure 4. Use $(x(0), y(0)) = (60, 70)$ and

- $(\xi(t), \eta(t)) = (8t, 0)$ for $t \in [0, 12]$ and $w = 10$,
- $(\xi(t), \eta(t)) = \begin{cases} (8t, 0), & t \in [0, 7) \\ (8(7-t), 0), & t \in [7, 12] \end{cases}$ and $w = 10$,
- $(\xi(t), \eta(t)) = (30 + 20\cos t, 20 + 15\sin t)$ and $t \in [0, 4\pi]$ and $w = 10$,
- $(\xi(t), \eta(t)) = (30 + 20\cos t, 20 + 15\sin t)$ and $t \in [0, 4\pi]$ and $w = 18$.

Solve the problem with other jogger's paths and dog speeds.

2.1.1 Error analysis of Euler's method

The analysis of Euler's method is useful to understand how it works, to predict the error when using it and perhaps to accelerate its convergence. Moreover, it gives an insight to how analyze other more efficient numerical methods.

We analyze the scaler IVP $y' = f(t, y)$ with $y(t_0) = y_0$ by assuming that it has a unique solution $y(t)$ on $t_0 \leqslant t \leqslant b$ and this solution has a bounded second derivative $y''(t)$ over this interval. Using the Taylor series formula we have

$$y(t_{k+1}) = y(t_k) + hy'(t_k) + \frac{1}{2}h^2 y''(\xi_k)$$

for some $t_k \leqslant \xi_k \leqslant t_{k+1}$. Using the fact that $y'(t) = f(t, y(t))$ we can write

$$y(t_{k+1}) = y(t_k) + hf(t_k, y(t_k)) + \frac{1}{2}h^2 y''(\xi_k). \tag{2.2}$$

The term

$$\tau_{k+1} = \frac{1}{2}h^2 y''(\xi_k)$$

is a *local truncation error* (or one-step error) for the Euler's method introduced in step $k+1$. Subtracting (2.2) from the Euler's rule

$$y_{k+1} = y_k + hf(t_k, y_k),$$

we will obtain

$$y(t_{k+1}) - y_{k+1} = (y(t_k) - y_k) + h[f(t_k, y(t_k)) - f(t_k, y_k)] + \tau_{k+1}, \tag{2.3}$$

which shows that the error in y_{k+1} consists of two parts:

- (1) the newly introduced local truncation error τ_{k+1},
- (2) the *propagated error* $(y(t_k) - y_k) + h[f(t_k, y(t_k)) - f(t_k, y_k)]$.

If we assume that

$$e_k = y(t_k) - y_k$$

then e_N would be the *global error* at the final time $t = t_N$. The global error reflects not only the local error at the final step, but also the compounded effects of the local errors at all previous steps. Unless in some specific situations where f is independent of y, **the global error is not simply the sum of the local errors.** For example, for ODE $y' = y$ since the solutions are diverging, the local errors at each step are magnified over time, so that the global error is greater than the sum of the local errors, as shown in Figure 8, where the local errors are indicated by small vertical bars between solutions and the global error is indicated by a bar at the end. On the other hand for ODE $y' = -y$ since the solutions are converging, the global error is less than the sum of the local errors. It is obvious that for the only case $f(t, y) = g(t)$, where the solutions are parallel curves, the global error is the direct sum of local errors. Because in this case we have $[f(t_k, y(t_k)) - f(t_k, y_k)] = g(t_k) - g(t_k) = 0$ and (2.3) reduced to $e_{k+1} = e_k + \tau_{k+1}$ for $k = 0, 1, \ldots, N-1$. We then simply have $e_N = \tau_1 + \tau_2 + \cdots + \tau_N$. However, for a general $f(t, y)$ we have to analyze the effect of $[f(t_k, y(t_k)) - f(t_k, y_k)]$ in each step. Considering $f(t, y)$ as a function of y and using the *mean value theorem*, we can write

$$f(t_k, y(t_k)) - f(t_k, y_k) = \frac{\partial f}{\partial y}(t_k, \eta_k)(y(t_k) - y_k)$$

for some η_k between $y(t_k)$ and y_k. Then, (2.3) yields

$$e_{k+1} = \left(1 + h\frac{\partial f}{\partial y}(t_k, \eta_k)\right) e_k + \tau_{k+1}, \tag{2.4}$$

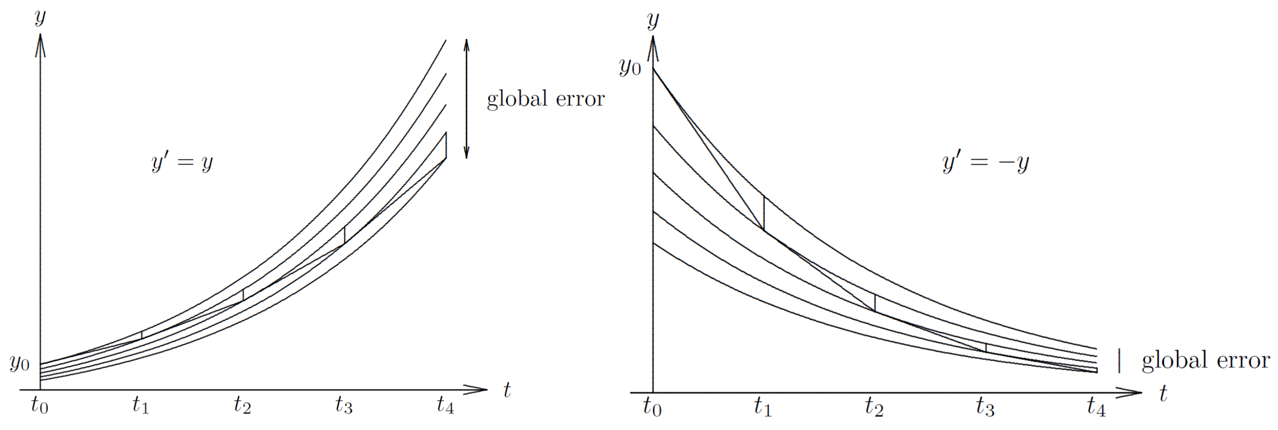


Figure 8: Local and global errors of Euler's method for $y' = y$ (left) and $y' = -y$ (right).

which shows that the amplification (or diminishing) factor for propagation error is $\left(1 + h\frac{\partial f}{\partial y}(t_k, \eta_k)\right)$ which is related to the stability of solutions via the Lipschitz constant of f.

We assume that the function $f(t, y)$ satisfies the following stronger Lipschitz condition: there exists a constant $L > 0$ such that

$$|f(t, y) - f(t, \tilde{y})| \leqslant L|y - \tilde{y}|, \quad \forall (t, y),\ (t, \tilde{y}) \in [t_0, b] \times \mathbb{R}.$$

In (2.4) if we go through absolute value of e_k, the term $\frac{\partial f}{\partial y}(t_k, \eta_k)$ can be replaced by the Lipschitz constant L to obtain

$$|e_{k+1}| \leqslant (1 + hL)|e_k| + |\tau_{k+1}|, \quad k = 0, 1, \ldots, N - 1 \tag{2.5}$$

from (2.4). We assume that

$$\tau(h) = \frac{1}{2}h\|y''\|_\infty = \frac{1}{2}h \max_{t_0 \leqslant t \leqslant b} |y''(t)|.$$

Then we have $|\tau_k| \leqslant h\tau(h)$ for $k = 1, \ldots, N$. Now, apply (2.5) recursively, we obtain

$$|e_k| \leqslant (1 + hL)^k|e_0| + [1 + (1 + hL) + (1 + hL)^2 + \cdots + (1 + hL)^{k-1}]h\tau(h).$$

Using the formula for sum of geometric series,

$$1 + r + r^2 + \cdots + r^{n-1} = \frac{r^n - 1}{r - 1}, \quad r \neq 1,$$

we obtain

$$|e_k| \leqslant (1 + hL)^k|e_0| + \left[\frac{(1 + hL)^k - 1}{L}\right]\tau(h).$$

Now we use the standard formula

$$1 + x \leqslant e^x, \quad x \geqslant 0$$

with $x = hL$ to obtain

$$|e_k| \leqslant e^{khL}|e_0| + \left[\frac{e^{khL} - 1}{L}\right]\tau(h).$$

On the other hand $kh = t_k - t_0$, so we can write

$$|e_N| \leqslant e^{L(t_N - t_0)}|e_0| + \left[\frac{e^{L(t_N - t_0)} - 1}{L}\right]\tau(h).$$

If $y_0 = y(t_0)$ or $|e_0| = |y(t_0) - y_0| \leqslant c_1 h$ then we have a global error of order h for the Euler's method, i.e.,

$$|e_N| \leqslant Ch, \quad C = c_1 e^{L(b-t_0)} + \frac{1}{2}\left[\frac{e^{L(b-t_0)}-1}{L}\right]\|y''\|_\infty. \tag{2.6}$$

Therefore, the Euler's method is said to converge with order 1. This order of convergence is obtained by assuming y to have a continuous second derivative y'' over interval $[t_0, b]$. When such assumption fails the error bound (2.6) no longer holds. See Workout 2.11.

Since the error bound (2.6) uses the Lipschitz constant L instead of $\frac{\partial f}{\partial y}$ and ignores the sign of $\frac{\partial f}{\partial y}$, it sometimes produces a very pessimistic numerical bound for the error. If

$$\frac{\partial f}{\partial y}(t,y) \leqslant 0 \tag{2.7}$$

then we may have a smaller than 1 amplification factor $\left(1 + h\frac{\partial f}{\partial y}(t_k, \eta_k)\right)$ instead of $(1+hL)$ which is always bigger than 1. In this case (negative partial derivative of f) if we assume that

$$L = \sup_{t\in[t_0,b], y\in\mathbb{R}} \left|\frac{\partial f}{\partial y}(t,y)\right|$$

and h is chosen so small that $1 - hL \geqslant -1$ then we have

$$1 \geqslant 1 + h\frac{\partial f}{\partial y}(t_k, \eta_k) \geqslant 1 - hL \geqslant -1,$$

and from (2.4) we can write

$$|e_{k+1}| \leqslant |e_k| + |\tau_k|, \quad k = 0, 1, \ldots, N-1.$$

Applying this bound recursively, we obtain

$$|e_N| \leqslant |e_0| + (b-t_0)\tau(h) = Ch, \quad C = c_1 + (b-t_0)\|y''\|_\infty \tag{2.8}$$

where $|e_0| \leqslant c_1 h$ is assumed. The constant C behind h in bound (2.8) is much smaller than that in bound (2.6) which contains the exponential terms. But, the error bound (2.8) is valid with restrictive assumption (2.7).

Example 2.2. For simple IVP

$$y'(t) = -y(t), \quad y(0) = 1, \quad 0 \leqslant t \leqslant b,$$

we have $\partial f(t,y)/\partial y = -1$ and $L = 1$. The true solution is $y(t) = e^{-t}$, hence $\|y''\|_\infty = 1$. With $y_0 = y(0) = 1$ ($|e_0| = 0$). From the bound (2.6) we have

$$|e_N| \leqslant \frac{1}{2}h(e^b - 1)$$

which shows the convergence with order h. However the constant in the bound grows exponentially in b. For example with $b = 5$ the bound becomes approximately $73.7h$ which is far larger than the actual error in Table 1. The error bound (2.8), on the other hand gives

$$|e_N| \leqslant \frac{1}{2}bh$$

which is very close to the actual error with different h in Table 1. The fifth column of the table also confirms the theoretical order 1 for the method.

Table 1: Euler's method: numerical solutions, errors, orders, and error bounds for IVP $y' = -y$ with $y_0 = 1$ at time $t = b = 5$.

h	y_N	$\lvert e_N \rvert$	$\lvert e_N \rvert / \lvert y(b) \rvert$	order	$\frac{1}{2}h(e^b - 1)$	$\frac{1}{2}bh$
0.2	3.778e − 3	2.960e − 3	4.393e − 1	–	14.74	0.5
0.1	5.154e − 3	1.584e − 3	2.351e − 1	0.90	7.37	0.25
0.05	5.921e − 3	8.174e − 4	1.213e − 1	0.95	3.69	0.12
0.025	6.323e − 3	4.149e − 4	6.158e − 2	0.98	1.84	0.06
0.0125	6.529e − 3	2.090e − 4	3.102e − 2	0.99	0.92	0.03
0.00625	6.633e − 3	1.049e − 4	1.557e − 2	0.99	0.46	0.02

The results of this table are obtained by executing the following code:

```
1   ExactSol = @(t) exp(-t);
2   b = 5;
3   h = 0.2;
4   for n = 1:6
5       [t,y] = ExEuler(@(t,y) -y, 1, [0 b], h);
6       AppSol(n) = y(end);
7       ABSerr(n) = abs(AppSol(n)-ExactSol(b));
8       RELerr(n) = ABSerr(n)/ExactSol(b);
9       h = h/2;
10  end
11  Order = log2(ABSerr(1:5)./ABSerr(2:6));
12  fprintf('y_N = \n'); fprintf('  %1.3e\n',AppSol);
13  fprintf('abs_err = \n'); fprintf('  %1.3e\n',ABSerr);
14  fprintf('rel_err = \n'); fprintf('  %1.3e\n',RELerr);
15  fprintf('order = \n'); fprintf('  %1.2f\n',Order);
```

Try understanding why the numerical orders are computed using that logarithmic formula in line 11 of the script above!

Table 2: Euler's method: numerical solutions, errors, orders, and error bounds for IVP $y' = +y$ with $y_0 = 1$ at time $t = b = 5$

h	y_N	$\lvert e_N \rvert$	$\lvert e_N \rvert / \lvert y(b) \rvert$	order	$\frac{1}{2}h(e^b - 1)\lVert y'' \rVert_\infty$
0.2	9.540e + 1	5.302e + 1	3.572e − 1	–	2.187e + 3
0.1	1.174e + 2	3.102e + 1	2.090e − 1	0.77	1.093e + 3
0.05	1.315e + 2	1.691e + 1	1.140e − 1	0.88	5.470e + 2
0.025	1.396e + 2	8.849e + 0	5.963e − 2	0.93	2.735e + 2
0.0125	1.439e + 2	4.529e + 0	3.052e − 2	0.97	1.367e + 2
0.00625	1.461e + 2	2.291e + 0	1.544e − 2	0.98	6.837e + 1

Now consider the IVP

$$y'(t) = y(t), \quad y(0) = 1, \quad 0 \leqslant t \leqslant b.$$

For this problem the the error bound (2.8) is not applicable because $\partial f(t,y)/\partial y = 1 > 0$. However, the error bound (2.6) is nearly sharp for this IVP. The exact solution is $y(t) = e^t$ and $\|y''\|_\infty = e^b$. See the results in Table 2.

Lab Exercise 2.3. Solve the following problems using Euler's method with stepsizes $h = 0.2, 0.1, 0.05, 0.025, 0.0125, 0.00625$. Compute the relative errors using the true solutions $y(t)$. In each case plot the error function in terms of h in the log-log scale, and compute the computational order of convergence.

- (a) $y'(t) = te^{-t} - y(t),\ 0 \leqslant t \leqslant 10,\ y(0) = 1$, with exact solution $y(t) = (1 + 0.5t^2)e^{-t}$.
- (b) $y'(t) = [\cos(y(t))]^2,\ 0 \leqslant t \leqslant 10,\ y(0) = 0$, with exact solution $y(t) = \tan^{-1}(t)$.
- (c) $y'(t) = t^3/y(t),\ 0 \leqslant t \leqslant 10,\ y(0) = 1$, with exact solution $y(t) = \sqrt{0.5t^4 + 1}$.

2.2 General explicit one-step methods

The Euler's method is a one-step method (counterpoise to multistep methods) as in each time level the approximate solution is obtained from merely the previous time level. A general explicit one-step method has the form

$$y_{k+1} = y_k + h\psi(t_k, y_k, h) \tag{2.9}$$

for a more general function ψ instead of f. We will assume that $\psi(t, y, h)$ is continuous in t and h and Lipschitz continuous in y, with Lipschitz constant $\tilde{L}$ that is generally related to the Lipschitz constant L of f.

Example 2.3. The choice $\psi(t, y, h) = f(t, y + \frac{h}{2}f(y))$ results in the two-stage *Runge-Kutta method* that will be addressed later. For this scheme we can simply show that ψ has Lipschitz constant $\tilde{L} = L + \frac{h}{2}L^2$ where L is the Lipschitz constant of f.

A one-step method is said to be *consistent* if

$$\psi(t, y, 0) = f(t, y), \tag{2.10}$$

for all t, y, and ψ is continuous in h. The consistency, indeed, implies that the local truncation error of method (2.9) is at least of order h^2, because

$$\begin{aligned}
\tau_{k+1} &= y(t_{k+1}) - y(t_k) - h\psi(t_k, y_k, h) \\
&= hy'(t_k) - h\psi(t_k, y(t_k), h) + \mathcal{O}(h^2) \\
&= h\psi(t_k, y(t_k), 0) - h[\psi(t_k, y(t_k), 0) + \mathcal{O}(h)] + \mathcal{O}(h^2) \\
&= \mathcal{O}(h^2)
\end{aligned}$$

The error analysis of general one-step methods can be obtained in a similar way as was done for the Euler's method. First, like as (2.2), the truncation error is obtained as

$$\tau_{k+1} = y(t_{k+1}) - y(t_k) - h\psi(t_k, y(t_k), h),$$

and then (2.3) is modified to

$$e_{k+1} = e_k + h[\psi(t_k, y(t_k), h) - \psi(t_k, y_k, h)] + \tau_{k+1}.$$

The reminder parts of analysis follow a same direction only L should be replaced by $\tilde{L}$ in new error bounds. We then can conclude the following theorem.

Theorem 2.4. If $\psi(t, y, h)$ is continuous in all its arguments and is Lipschitz continuous in its second argument, and the consistency condition (2.10) holds, then the explicit one-step method (2.9) is convergent with a global error of at least order h^1. If the local truncation errors τ_{k+1} behave as h^{p+1}, then the global order is of order h^p.

2.3 Zero-stability

In the convergence proof of the one-step methods we observed the effect of amplification factor in propagating the local errors which finally was summed up to factor C in the error bound (2.6). Although this factor grows in b, it is bounded independent of h as $h \to 0$. Consequently the method is stable. This form of stability for a numerical method is often called **zero-stability**, since it is concerned with the stability of the method in the limit as h tends to zero. To see this observation in a more relevant presentation to stability of the original IVP (1.4), assume that the initial condition y_0 is perturbed by ε and define numerical solutions of the perturbed problem by

$$z_{k+1} = z_k + hf(t_k, z_k), \quad z_0 = y_0 + \varepsilon.$$

For comparing two numerical solutions y_k and z_k, let $e_k = z_k - y_k$. Then $e_0 = \varepsilon$ and subtracting from $y_{k+1} = y_k + hf(t_k, y_k)$ we obtain

$$e_{k+1} = e_k + h[f(t_k, z_n) - f(t_k, y_k)].$$

This is exactly the same form as (2.3) with τ_{k+1} set to be zero. Using the same procedure we obtain

$$|e_{k+1}| \leqslant (1 + hL)^k |e_0| \leqslant e^{L(t_k - t_0)} |\varepsilon|.$$

Consequently, we can write

$$\max_{0 \leqslant k \leqslant N} |z_k - y_k| \leqslant e^{L(b-t_0)} |\varepsilon|$$

which is the analog to the stability result (1.14) for the original IVP (1.4). The *zero-stability* is different from other forms of stability which are of equal importance in practice. The fact that a method is zero-stable (and converges as $h \to 0$) is no guarantee that it will give reasonable results on the particular grid with $h > 0$ that we want to use in practice. Other stability issues of a different nature will be taken up in the next sections.

2.4 Absolute stability

The zero-stability is needed to guarantee convergence of a numerical method as $h \to 0$. In practice, however, we need to perform a single calculation using a given positive stepsize $h > 0$. Moreover, to minimize the computational cost a larger as possible h (consistent with our desired accuracy) is usually preferred. A stronger form of stability than the zero-stability is required in this case to force the method to work for this particular stepsize h. Let's illustrate the situation in three numerical examples borrowed from [**LeVeque:2007**].

Example 2.4. We are going to apply the Euler's method on a simple IVP of the form

$$y'(t) = -\sin(t), \quad 0 \leqslant t \leqslant 2, \quad y(0) = 1$$

with exact solution $y(t) = \cos t$. Since $f(t, y) = \sin(t)$ is independent of y, $(L = 0)$ the global error is the sum of local errors

$$|\tau_k| = \frac{h^2}{2}|y''(\xi_k)| \leqslant \frac{h^2}{2}.$$

Indeed we have

$$|e_N| \leqslant (b - t_0)\tau(h) = 2\tau(h) = h.$$

Suppose we want to compute the solution at $t = 2$ with a global error less than 0.001. According to the error bound it suffices to take $h = 0.001$ and obtain the approximate solution after 2000 time steps. The computed solution $y_{2000} \doteq -0.4156921$ has error $|e_{2000}| = |y_{2000} - \cos(2)| \doteq 0.45 \times 10^{-3}$.

Now, we change the IVP to

$$y'(t) = \lambda(y - \cos t) - \sin(t), \quad 0 \leqslant t \leqslant 2, \quad y(0) = 1, \tag{2.11}$$

for some constant λ. The exact solution is $y(t) = \cos t$, as before. Let $\lambda = -10$. The error bound (2.8) suggests again the global error $|e_N| \leqslant h$. For this reason, we again choose $h = 0.001$ for a global error less than 0.001. The computed solution now is $y_{2000} \doteq -0.4161629$ with error $|e_{2000}| \doteq 0.16 \times 10^{-4}$ which is even better than the previous one.

Let us examine some larger (in magnitude) λ. Let $\lambda = -2100$. Executing the Euler's method gives $y_{2000} \doteq 0.15 \times 10^{77}$ which is far away from the exact solution and shows a blown up in computations. The method is zero-stable and we proved that when $h \to 0$ it is convergent. Indeed, for sufficiently small stepsizes we achieve accurate results as reported in Table 3.

Table 3: Global errors for the Euler's method with different stepsizes.

h	0.001	0.00097	0.00095	0.0008	0.0004
$\lvert e_N \rvert$	0.15e + 77	0.77e + 26	0.40e − 07	0.79e − 07	0.40e − 08

Something dramatic happens for values of h between 0.00095 and 0.00097. For smaller values of h we get very good results, whereas for larger values of h the solution blows up. To find the reason, we come back to (2.4) where we have for the linear IVP with

$f(t, y) = \lambda(y - \cos t) - \sin(t)$ the recursion

$$e_{k+1} = (1 + \lambda h)e_k + \tau_{k+1}.$$

This means that in each time step the previous error is multiplied by factor $(1 + \lambda h)$. With $\lambda = -2100$ and $h = 0.001$ we have $|1 + \lambda h| = 1.1$. After 2000 steps the truncation error introduced in the first step has grown by a factor of roughly $(1.1)^{2000} \approx 10^{82}$, which is consistent with the error actually seen. Note that with $\lambda = 10$, we have $|1 + \lambda h| = 0.99$ causing a decay in the effect of previous errors in each step. For the first case, i.e., $\lambda = 0$, the amplification factor is 1 the reason why we got a worse result in this case than the case of $\lambda = -10$.
Consequently, we can argue that for values of h satisfying

$$|1 + \lambda h| \leqslant 1$$

the Euler method produces stable and accurate results for IVP (2.11). In the case of $\lambda = -2100$ the above criterion suggests the values of h smaller than $2/2100 \doteq 0.000952$.

Remark 2.1. Note that the exponential growth of errors for some positive values of h in the previous example does not contradict zero-stability or convergence of the method in any way. The method does converge as $h \to 0$.

Example 2.4 shows that another notion of stability is needed to force a numerical method to produce stable and accurate results with a given step length $h > 0$. There exists a wide variety of "stability" notions but one that is most basic is the **absolute stability**. This kind of stability is based on the linear *test equation*

$$y'(t) = \lambda y(t), \quad \lambda \in \mathbb{C}. \tag{2.12}$$

The restriction on the step length $h > 0$ on which the method will work for test ODE (2.12) is called the absolute stability conditions. For example, the Euler's method if is applied on (2.12) yields

$$y_{k+1} = (1 + \lambda h)y_k = (1 + \lambda h)^2 y_{k-1} = \cdots = (1 + \lambda h)^{k+1} y_0.$$

In order to prevent a blown up in the solution when $k \to \infty$ we should impose the condition

$$|1 + \lambda h| \leqslant 1.$$

If $\lambda \in \mathbb{R}$, the stability condition will be $-2 \leqslant z \leqslant 0$ for $z \equiv \lambda h$. This implies that

$$\lambda \leqslant 0 \text{ and } 0 \leqslant h \leqslant \frac{2}{-\lambda}.$$

In the general case $\lambda \in \mathbb{C}$, the complex number z should satisfy $|1 + z| \leqslant 1$ which means that z should lie inside and on a circle with center $(-1, 0)$ and radius 1 in the complex plane. This region is called the **region of absolute stability** of the Euler's method. See shadow part on the left hand side of Figure 9.

Definition 2.5. By applying a one-step method on the test problem (2.12) we get

$$y_{k+1} = R(z)y_k, \quad z = \lambda h$$

for some function $R(z)$, and then the absolute stability region of the method is defined to be

$$S = \{z \in \mathbb{C} : |R(z)| \leqslant 1\}.$$

The stability region of the explicit Euler's method is rather small compared to other (usually) implicit methods. This will impose a serious restriction on stepsize h to guarantee the convergence. This restriction together with slow convergence rate of Euler's method convince us to study and develop other more efficient ODE solvers. For comparison, the absolute stability region of the implicit Euler's method (see the next section) is drown on the right hand side of Figure 9. It contains all the complex plane except a unit circle with center $(1, 0)$.

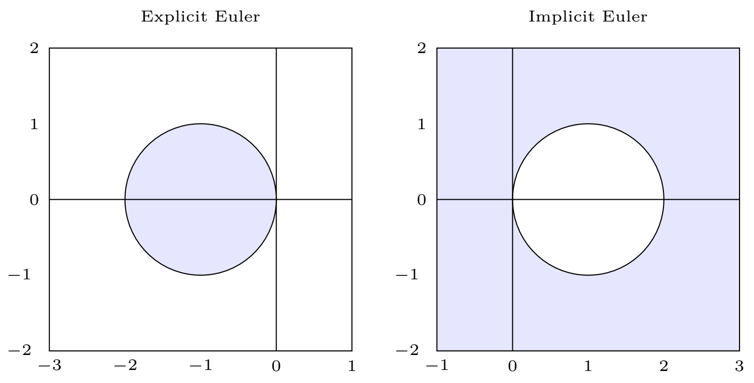


Figure 9: Absolute stability regions of explicit and implicit Euler's methods

Although the absolute stability region is determined by testing the method on simple linear ODE (2.12), it yields information that is typically useful in determining an appropriate step length in nonlinear problems as well.

For a system of ODEs of the form

$$\boldsymbol{y}'(t) = A\boldsymbol{y}(t), \quad A \in \mathbb{R}^{n\times n} \tag{2.13}$$

where A is diagonalizable with eigenvalues λ_ℓ, $\ell = 1, 2, \ldots, n$, a numerical method is absolutely stable if $z_\ell = \lambda_\ell h$ all lie in the absolute stability region of the method in the scaler case. The proof is simple, because as we observed in (1.13) the system can be decoupled to n scaler ODE

$$u_\ell' = \lambda_\ell u_\ell, \quad \ell = 1, 2, \ldots, n.$$

Now, we investigate a numerical solution of a simple partial differential equation (PDE) using the method of lines (MOL) which results in a linear system of ODEs.

Example 2.5. Consider the linear *diffusion* equation

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2}, \quad 0 \leqslant x \leqslant 1, \quad t \geqslant 0$$

with homogeneous Dirichlet boundary conditions $u(0,t) = u(1,t) = 0$ and initial condition $u(x,0) = u^0(x)$. The method of lines (MOL) solution if is applied on this problem with the central difference approximation

$$\frac{\partial^2 u}{\partial x^2}(x_k,t) \approx \frac{u(x_{k+1},t) - 2u(x_k,t) + u(x_{k-1},t)}{(\Delta x)^2}, \quad \Delta x = \frac{1}{m+1},$$

with $x_k = k\Delta x$, leads to a system of equations of the form (2.13) with

$$\boldsymbol{y}(t) = \begin{bmatrix} u(x_1,t) \\ u(x_2,t) \\ \vdots \\ u(x_{m-1},t) \\ u(x_m,t) \end{bmatrix}, \quad A = \frac{1}{(\Delta x)^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}.$$

The matrix A is symmetric and tridiagonal. There exists a close formula for its eigenvalues:

$$\lambda_\ell = \frac{2}{(\Delta x)^2}(\cos(\pi\ell\Delta x) - 1) = \frac{-4}{(\Delta x)^2}\sin^2(\frac{\pi}{2}\ell\Delta x), \quad \ell = 1,\ldots,m.$$

The distribution of eigenvalues for two different matrix size $m = 10, 20$ (or $\Delta x = 1/11, 1/21$) are displayed in Figure 10.

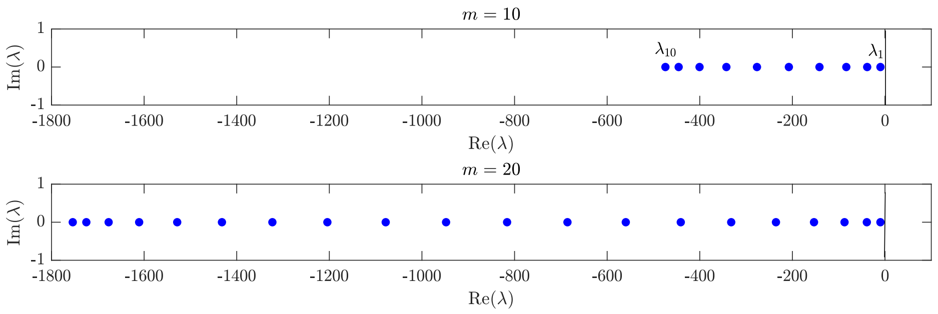


Figure 10: Distribution of eigenvalues of matrix A.

All eigenvalues are real (because A is symmetric) and fall on the left-half (complex) plane. If one insists to apply the explicit Euler's method for solving this system then the step length h should be chosen small enough such that all $\lambda_\ell h$ lie in the absolute stability region of the method. Since the largest (in magnitude) eigenvalue is λ_m, the absolute stability is guaranteed if the step length is chosen equal to or less than

$$\frac{2}{-\lambda_m} = \frac{(\Delta x)^2}{2\sin^2(\frac{\pi}{2}m\Delta x)}$$

Since $\sin^2(\frac{\pi}{2}m\Delta x) < 1$, it is enough to take the step length equal to or less than $\frac{1}{2}(\Delta x)^2$. This is a serious restriction for numerical solution of such PDE.

Remark 2.2. A method is zero-stable if the origin belongs to its region of absolute stability.

2.5 Implicit methods

Euler's method is an explicit method in that it uses the already known information at time t_k to advance the solution to time t_{k+1}. However, this method has a rather small stability region.

The **implicit Euler's method** (backward Euler's method) is obtained by approximating $y'(t_k)$ by the first-order backward difference approximation

$$y'(t_k) = \frac{y(t_k) - y(t_{k-1})}{h} - \frac{h}{2}y''(\xi_k), \quad t_{k-1} \leqslant \xi_k \leqslant t_k.$$

By dropping the error term and using the approximate values y_k instead of $y(t_k)$, we obtain an algebraic equation $y_k = y_{k-1} + hf(t_k, y_k)$ for $k = 1, 2, \ldots$. Shifting the index by 1, the implicit Euler's method is obtained as

$$\begin{aligned} &y_{k+1} = y_k + hf(t_{k+1}, y_{k+1}), \quad k = 0, 1, \ldots, N-1, \\ &y_0 = y(t_0). \end{aligned} \tag{2.14}$$

This scheme is implicit because we must evaluate f with the argument y_{k+1} before we know its value. If f is a nonlinear function in y then a rootfinding method such as fixed-point iteration or Newton's method can be used. A good starting guess for the iteration is the solution at the previous time step or one step solution of the explicit Euler's method. If f is Lipschitz continuous and h is small enough it can be proved that $y - y_k + hf(t_{k+1}, y) = 0$ has a unique root.

Usually, a simple iteration method is efficient for solving the nonlinear equation in each step. In step $k+1$, given an initial guess $y_{k+1}^{(0)}$, we define $y_{k+1}^{(1)}, y_{k+1}^{(2)}, \ldots$ by

$$y_{k+1}^{(j+1)} = y_k + hf(t_{k+1}, y_{k+1}^{(j)}), \quad j = 0, 1, 2, \ldots. \tag{2.15}$$

Subtracting (2.15) from (2.14), we obtain

$$y_{k+1} - y_{k+1}^{(j+1)} = h\left[f(t_{k+1}, y_{k+1}) - f(t_{k+1}, y_{k+1}^{(j)})\right].$$

If we assume that f is Lipschitz continuous with constant L then we can write

$$|y_{k+1} - y_{k+1}^{(j+1)}| \leqslant hL|y_{k+1} - y_{k+1}^{(j)}|.$$

This means that if h is chosen small enough such that

$$hL \leqslant 1 \tag{2.16}$$

then the error will converge to zero for a sufficiently good initial guess $y_{k+1}^{(0)}$. In practice, usually one step of the explicit Euler's method, i.e.,

$$y_{k+1}^{(0)} = y_k + hf(t_k, y_k)$$

is used as an initial guess in each step of the implicit Euler's method. This is called a *predictor formula* as predicts the root of the implicit method. Besides, h is chosen so small such that (2.16) is much smaller than 1 to have a rapid fixed-point convergence. Often few iterates (sometimes only one iterate) need(s) to obtain a satisfactory result.

Another practical way is to assume $y_{k+1}^{(0)} = y_k$ and do the iteration (2.15) twice. This two-point iteration is equivalent to following two-step scheme

$$\begin{aligned} z &= y_k + hf(t_{k+1}, y_k) \\ y_{k+1} &= y_k + hf(t_{k+1}, z), \end{aligned}$$

or, by writing it in a one line,

$$y_{k+1} = y_k + hf(t_{k+1}, y_k + hf(t_{k+1}, y_k)). \tag{2.17}$$

This method is still of first-order accuracy but has some absolute stability limitations. However, the implicit Euler's method (2.14) if is applied on test equation (2.12) gives

$$y_k = \frac{1}{(1-\lambda h)^k} y_0.$$

The instability never happens if

$$\frac{1}{|1-\lambda h|} \leqslant 1.$$

Therefore, the region of absolute stability of the method is $S = \{z \in \mathbb{C} : |1-z| \geqslant 1\}$ which is shown on the right hand side of Figure 9.

One drawback of both explicit and implicit Euler's methods is the low convergence order. Before presenting new higher order schemes let us discuss another approach for obtaining Euler's formulas. If we integrate the equation $y'(t) = f(t, y(t))$ from t_k to t_{k+1}, we obtain

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(\tau, y(\tau))d\tau. \tag{2.18}$$

The explicit Euler's method will be resulted if the integral in (2.18) is approximated by the *box rule*

$$\int_a^b g(\tau)d\tau \approx (b-a)g(a)$$

while the implicit Euler's method follows from the box quadrature

$$\int_a^b g(\tau)d\tau \approx (b-a)g(b).$$

More accurate quadratures can be used to obtain more accurate methods. For instance, we can implement the *trapezoidal rule* (with the error term)

$$\int_a^b g(\tau)d\tau = \frac{1}{2}(b-a)[g(a)+g(b)] - \frac{1}{12}(b-a)^3 g''(\xi), \tag{2.19}$$

for some $a \leqslant \xi \leqslant b$. Applying (2.19) to (2.18), we obtain

$$y(t_{k+1}) = y(t_k) + \frac{h}{2}[f(t_k, y(t_k)) + f(t_{k+1}, y(t_{k+1}))] - \frac{h^3}{12} y^{(3)}(\xi_k) \tag{2.20}$$

for some $t_k \leqslant \xi_k \leqslant t_{k+1}$. The second derivative in the error term of the trapezoidal rule (2.19) is replaced by the third derivative of y in (2.20) because $f(t,y) = y'(t)$. By dropping the error term in (2.20) and replacing $y(t_k)$ by approximate values y_k, the **trapezoidal method** is obtained as

$$\begin{aligned} & y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k) + f(t_{k+1}, y_{k+1})], \quad k = 0, 1, 2, \dots \\ & y_0 = y(t_0). \end{aligned} \tag{2.21}$$

The local truncation error for this method is

$$\tau_{k+1} = -\frac{h^3}{12} y^{(3)}(\xi). \tag{2.22}$$

It can be proved that the trapezoidal method is of second-order accuracy and its global error satisfies

$$|e_N| \leqslant Ch^2$$

for all sufficiently small h. The proof follows the same sketch as the proof of the explicit Euler's method. In additions, we can simply show that the region of absolute stability of the trapezoidal method is the left half plane as shown in Figure 11.

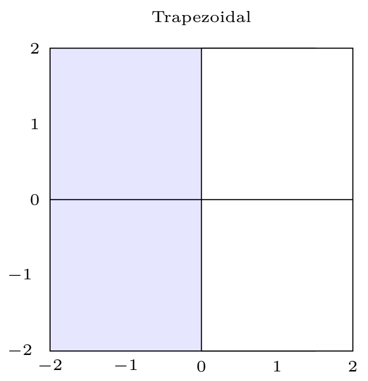


Figure 11: Absolute stability region of the trapezoidal method

Workout 2.6. Show that the region of absolute stability of the trapezoidal method is the left half complex plane.

The convergence order 2 and the absolute stability of the trapezoidal method are two advantages that make this method an important tool for solving ordinary differential equations.

When $f(t, y)$ is nonlinear in y, the discussion for the solution of the implicit Euler's method applies to the solution of the trapezoidal method (2.21) with a slight variation. The iteration formula (2.15) is replaced by

$$y_{k+1}^{(j+1)} = y_k + \frac{h}{2}[f(t_k, y_k) + f(t_{k+1}, y_{k+1}^{(j)}), \quad j = 0, 1, 2, \dots. \tag{2.23}$$

The convergence condition (2.16) is replaced by

$$\frac{hL}{2} \leqslant 1.$$

The usual choice of the initial guess $y_{k+1}^{(0)}$ for (2.23) is the Euler's solution

$$y_{k+1}^{(0)} = y_k + hf(t_k, y_k).$$

With this choice the resulting global error will be still of order h^2, because a one step error of the Euler's method is of order h^2. If only one iteration of (2.23) is used the resulting new scheme is

$$y_{k+1} = y_k + \frac{h}{2}[f(t_k, y_k) + f(t_{k+1}, y_k + hf(t_k, y_k))], \tag{2.24}$$

which is also known as *Heun's method.* This method is still of second order accuracy, but with a more restricted region of absolute stability. The Heun's method is identical with one of the

Runge-Kutta methods of order two. We will address various types of Runge-Kutta method in a forthcoming section.

Workout 2.7. Show that the absolute stability regions of schemes (2.17) and (2.24) are bounded. Especially both of them do not include the whole negative real line.

Workout 2.8. Let $\theta \in [0,1]$ and consider the θ-method

$$y_{k+1} = y_k + h[(1-\theta)f(t_k, y_k) + \theta f(t_{k+1}, y_{k+1})].$$

(a) Which values of θ correspond to the explicit Euler, implicit Euler, and trapezoidal methods? (b) Separate two cases $\theta \in [0, 1/2)$ and $\theta \in [1/2, 1]$. In which case the left-half plane lies in the absolute stability regions of this method?
Optional: answer the above questions for the *generalized midpoint* method

$$y_{k+1} = y_k + hf(t_{k+\theta}, (1-\theta)y_k + \theta y_{k+1})$$

where $t_{k+\theta} = (1-\theta)t_k + \theta t_{k+1}$.

Workout 2.9. Consider the scheme

$$y_{k+1} = y_{k-1} + 2hf(t_k, y_k),$$

for solving $y'(t) = f(t, y)$. Show that the local truncation error of this scheme is of order h^3. The absolute stability region for this scheme is $S = \{z = \alpha + i\beta \in \mathbb{C} : \alpha = 0, -1 \leqslant \beta \leqslant 1\}$, which is a marginal stability region (no need to prove!) This method is known as *midpoint* or *leapfrog method*. Is the leapfrog method A-stable?
Hint: To obtain the truncation error, replace y_k by exact values $y(t_k)$ and leave the formula by additional term τ_k. Then use Taylor expansion to determine τ_k.

Lab Exercise 2.10. Write the MATLAB functions for implicit and trapezoidal methods. In each case, use the iterative method and/or a predictor formula for nonlinear IVPs. Repeat solving examples in Lab Exercise 2.3 using implicit Euler and trapezoidal methods and compare the errors and orders. Comment on results.

Lab Exercise 2.11. Consider the IVP

$$y'(t) = \frac{1}{t}y(t) + (\alpha-1)t^{\alpha-1}, \quad y(0) = 0, \quad t \in [0,1],$$

with $\alpha > 0$. The solution is $y(t) = t^\alpha$. To have y twice continuously differentiable, we need $\alpha \geqslant 2$. Use your MATLAB codes for explicit and implicit Euler and trapezoidal methods for $\alpha = 2.5, 1.5, 1.1$ with stepsizes $h = 0.2, 0.1, 0.05, 0.025, 0.0125$. Determine the computational convergence orders. Compare with theoretical orders and report a reason for your observation.

2.6 Taylor series methods

Euler's methods can be formulated by using a Taylor series approximation when y' is replaced by $f(t,y)$ and higher order derivative in the series are dropped. One can of course use higher order terms but then y'', y''', … should be obtained by differentiating the differential equation

$$y'(t) = f(t,y),$$

successively. From the Taylor series expansion of order p we have

$$y(t_{k+1}) \approx y(t_k) + hy'(t_k) + \frac{h^2}{2!}y''(t_k) + \cdots + \frac{h^p}{p!}y^{(p)}(t_k) \tag{2.25}$$

where the truncation error is

$$\tau_{k+1} = \frac{h^{p+1}}{(p+1)!}y^{(p+1)}(\xi_k), \quad t_k \leqslant \xi_k \leqslant t_{k+1}. \tag{2.26}$$

The term $y'(t)$ in (2.26) can be replaced by $f(t,y)$ as we have done in Euler's methods. For higher order derivatives we can write

$$\begin{aligned} y''(t) &= f_t + f_y f \\ y^{(3)}(t) &= f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_y(f_t + f_y f) \\ &\vdots \end{aligned}$$

provided that partial derivatives of $f(t,y)$ with respect to y exist. Substituting these formulas into (2.25), we obtain

$$y_{k+1} = y_k + hy'_k + \frac{h^2}{2}y''_k + \cdots + \frac{h^p}{p!}y_k^{(p)}, \tag{2.27}$$

which is called the **Taylor series method**. The derivatives formulas in (2.27) are

$$y'_k = f(t_k, y_k), \quad y''_k = (f_t + f_y f)(t_k, y_k), \text{ and so on.}$$

The formulas for higher order derivatives rapidly become too complicated, so Taylor series methods of higher order have not often been used in practice. Recently, however, the availability of symbolic manipulation and automatic differentiation systems have made these methods more feasible.

If the solution y and the derivative function $f(t,y)$ are sufficiently differentiable then it can be proved that the global error for the scheme (2.27) satisfies

$$|e_N| \leqslant Ch^p \|y^{(p+1)}\|_\infty,$$

which means that the method is of p-th order accuracy. The constant C is something similar to that was obtained for the explicit Euler's method (the first order Taylor method).

Lab Exercise 2.12. Construct the Taylor series methods of orders 2 and 3 for IVP

$$y'(t) = [\cos y(t)]^2, \quad 0 \leqslant t \leqslant 10, \quad y(0) = 0.$$

Write a MATLAB code to compute the results for stepsizes $h = 0.2$, 0.1,0.05, 0.025, 0.0125, 0.00625. Plot the error functions in each case and calculate numerical orders. Compare the results with Euler and trapezoidal methods. The exact solution of the above IVP is $y(t) = \tan^{-1}(t)$. Use this information to calculate errors and orders.

2.7 Runge-Kutta methods

The calculation of higher order partial derivatives of $f(t, y)$ makes the Taylor series methods complicated and time-consuming. **Runge-Kutta** methods (abbreviated by RK methods) replace higher derivatives by more evaluations of $f(t, y)$ to have finite difference approximations for derivatives while retain the accuracy of Taylor series methods. The RK methods are *one-step* but *multi-stage* and are fairly easy to program not only for a scaler ODE but also for a system of ODEs.

To derive a second order RK method, consider the second order Taylor method

$$y_{k+1} = y_k + hy'_k + \frac{h^2}{2} y''_k \tag{2.28}$$

where $y' = f(t, y)$ and $y'' = f_t + f_y f$ both evaluated at (t_k, y_k). We aim to approximate $f_t + f_y f$ by expanding f in a bivariate Taylor series as

$$f(t + h, y + hf) = \Big(f + hf_t + hff_y\Big)(t, y) + \mathcal{O}(h^2).$$

This simply shows that

$$y''(t) = (f_t + f_y f)(t, y) = \frac{1}{h}[f(t + h, y + hf(t, y)) - f(t, y)] + \mathcal{O}(h).$$

Dropping the $\mathcal{O}(h)$ term and substituting in (2.28), we obtain

$$\begin{aligned} y_{k+1} &= y_k + hf(t_k, y_k) + \frac{h^2}{2}\frac{1}{h}[f(t_k + h, y_k + hf(t_k, y_k)) - f(t_k, y_k)] \\ &= y_k + \frac{h}{2}[f(t_k, y_k) + f(t_k + h, y_k + hf(t_k, y_k))]. \end{aligned} \tag{2.29}$$

This method was previously derived as the Heun's method in (2.24). As an RK2 method it is usually written in the following two step pattern:

$$\begin{aligned} z_1 &= y_k \\ z_2 &= y_k + hf(t_k, z_1) \\ y_{k+1} &= y_k + \frac{h}{2}[f(t_k, z_1) + f(t_k + h, z_2)]. \end{aligned} \tag{2.30}$$

This is not the only order 2 explicit RK method. As we discussed in section 2.2, a general explicit method can be written as

$$y_{k+1} = y_k + h\psi(t_k, y_k, h), \quad y_0 = y(t_0).$$

In the RK2 method (2.29) we derived $\psi(t, y, h)$ as

$$\psi(t, y, h) = \frac{1}{2}f(t, y) + \frac{1}{2}f(t + h, y + hf(t, y)).$$

This formula can be generalized to ansatz

$$\psi(t, y, h) = b_1 f(t, y) + b_2 f(t + \alpha h, y + \beta h f(t, y)), \tag{2.31}$$

with unknown coefficients b_1, b_2, α, β that can be determined such that the local truncation error

$$\tau_{k+1} = y(t_{k+1}) - [y(t_k) + h\psi(t_k, y(t_k), h)]$$

will satisfy $\tau_{k+1} = \mathcal{O}(h^3)$ just as with the Taylor method of order 2. After some manipulations with the bivariate Taylor expansion, we will obtain the relations

$$b_2 \neq 0, \quad b_1 = 1 - b_2, \quad \alpha = \beta = \frac{1}{2b_2}.$$

between the coefficients in order to have $\tau_{k+1} = \mathcal{O}(h^3)$. Depending on the choice of b_2, there exists a family of RK methods of order 2. The case $b_2 = 1/2$ results in RK method (2.29). Another choice $b_2 = 1$, $b_1 = 0$ and $\alpha = \beta = \frac{1}{2}$, results in

$$y_{k+1} = y_k + hf(t_k + \tfrac{1}{2}h, y_k + \tfrac{1}{2}hf(t_k, y_k)). \tag{2.32}$$

or in a multi-stage format

$$\begin{aligned} z_1 &= y_k \\ z_2 &= y_k + \frac{h}{2} f(t_k, z_1) \\ y_{k+1} &= y_k + hf(t_k + \tfrac{h}{2}, z_2). \end{aligned} \tag{2.33}$$

A general explicit RK method is defined as below.

Definition 2.13. An explicit RK method with s stages has the form

$$\begin{aligned} z_1 &= y_k, \\ z_2 &= y_k + ha_{2,1} f(t_k, z_1), \\ z_3 &= y_k + h\Big[a_{3,1} f(t_k, z_1) + a_{3,2} f(t_k + c_2 h, z_2)\Big], \\ &\vdots \\ z_s &= y_k + h\Big[a_{s,1} f(t_k, z_1) + a_{s,2} f(t_k + c_2 h, z_2) + \cdots + a_{s,s-1} f(t_k + c_{s-1} h, z_{s-1})\Big], \\ y_{k+1} &= y_k + h\Big[b_1 f(t_k, z_1) + b_2 f(t_k + c_2 h, z_2) + \cdots + b_s f(t_k + c_s h, z_s)\Big]. \end{aligned} \tag{2.34}$$

Such RK method is fully determined by coefficients $\{c_\ell, a_{\ell,j}, b_j\}$. These coefficients are usually displayed in a table called **Butcher's Tableau**[2]

[2]After John Charles Butcher (1933-present) who is a New Zealand mathematician and an specialist in numerical methods for ODEs.

$$
\begin{array}{r|ccccc}
0 = c_1 & & & & & \\
c_2 & a_{2,1} & & & & \\
c_3 & a_{3,1} & a_{3,2} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}
$$

RK methods can be expressed in the general form (2.9) with

$$
\psi(t, y, h) = \sum_{j=1}^{s} b_j f(t + c_j h, z_j), \quad z_j = y + h \sum_{\ell=1}^{j-1} a_{j\ell} f(t + c_\ell h, z_\ell).
$$

The consistency condition $\psi(t, y, 0) = f(t, y)$ holds if

$$
\sum_{j=1}^{s} b_\ell = 1. \tag{2.35}
$$

According to Theorem 2.4, if f is continuous in t and Lipschitz continuous in y, and the condition (2.35) holds, then the RK method (2.34) is convergent.

Moreover, in a RK method we always assume that

$$
\sum_{j=1}^{\ell-1} a_{\ell j} = c_\ell, \quad \ell = 1, 2, \ldots, s, \tag{2.36}
$$

which ensure that intermediate values z_ℓ provide approximations of order at least 1 to exact values $y(t_k + c_\ell h)$. Conditions (2.36) are called the *stage conditions* for RK methods.

Example 2.6. The Butcher's tableau of the RK2 method (2.30) is

$$
\begin{array}{c|cc}
0 & & \\
1 & 1 & \\
\hline
 & 1/2 & 1/2
\end{array}
$$

while the RK2 method (2.33) has a Butcher's tableau of the form

$$
\begin{array}{c|cc}
0 & & \\
1/2 & 1/2 & \\
\hline
 & 0 & 1
\end{array}
$$

There also exist a family of third-order RK methods. The Butcher's tableau of one of these schemes is:

$$
\begin{array}{r|ccc}
0 & & & \\
1/2 & 1/2 & & \\
1 & -1 & 2 & \\
\hline
 & 1/6 & 2/3 & 1/6
\end{array}
$$

Workout 2.14. (a) Convert the above tableau into a 3-stage RK formula. (b) Search the internet to find another RK3 method and write down its stages.

A very popular explicit RK method is the following fourth-order scheme (RK4):

$$
\begin{aligned}
z_1 &= y_k,\\
z_2 &= y_k + \tfrac{1}{2}hf(t_k, z_1),\\
z_3 &= y_k + \tfrac{1}{2}hf(t_k + \tfrac{1}{2}h, z_2),\\
z_4 &= y_k + hf(t_k + \tfrac{1}{2}h, z_3),\\
y_{k+1} &= y_k + \frac{h}{6}\Big[f(t_k, z_1) + 2f(t_k + \tfrac{1}{2}h, z_2) + 2f(t_k + \tfrac{1}{2}h, z_3) + f(t_k + h, z_4)\Big].
\end{aligned}
\tag{2.37}
$$

The Butcher's tableau for this method is

$$
\begin{array}{c|cccc}
0 & & & & \\
1/2 & 1/2 & & & \\
1/2 & 0 & 1/2 & & \\
1 & 0 & 0 & 1 & \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}
$$

Using a similar but more tedious calculation than that was done for method (2.30), we can show that the local truncation error for this 4-stage method is of order h^5. Then by applying the sketch given in section 2.2 for analysing general explicit methods we can show that the global error of method (2.37) satisfies

$$|e_N| \leqslant Ch^4$$

which shows that this method is of fourth-order accuracy, for this reason is usually called RK4. This is not the only fourth-order explicit RK method; there exists a family of such methods with different Butcher's tableaus.

Workout 2.15. Show that the RK4 method (2.37) when is applied on simple differential equation $y'(t) = f(t)$ (with no dependence of f on y) reduces to Simpson's rule for numerical integration.

Like as other explicit methods, RK methods have restricted regions of absolute stability. For example, by applying the RK2 formula (2.29) on test equation $y'(t) = \lambda y$, we obtain

$$y_k = (1 + \lambda h + \frac{1}{2}(\lambda h)^2)y_k,$$

which shows that the region of absolute stability for this method is

$$S = \{z \in \mathbb{C} : |1 + z + \tfrac{1}{2}z^2| \leqslant 1\}.$$

This region is shown in Figure 12 (left panel). The stability regions for an RK3 method and the RK4 method (2.37), obtained in a similar way, are shown in the middle and right sides of Figure 12.

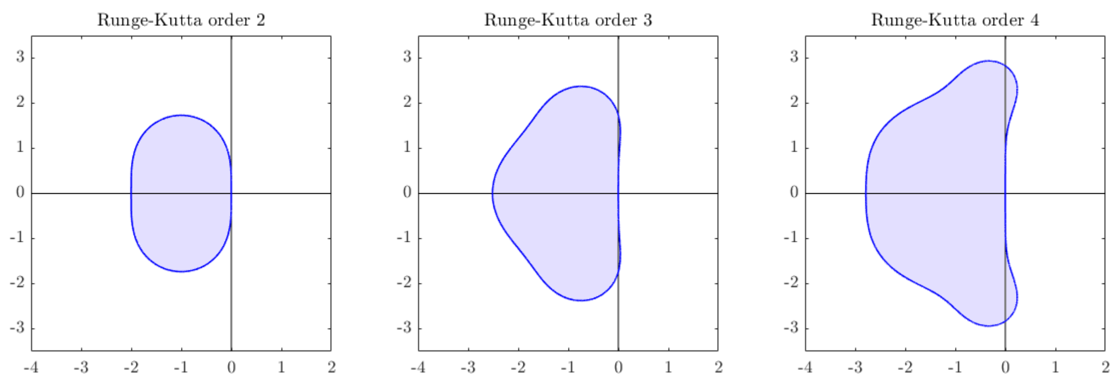


Figure 12: Absolute stability regions of Runge-Kutta methods of orders 2, 3 and 4.

Lab Exercise 2.16. Use a MATLAB code for solving the IVP

$$y'(t) = \frac{1}{1+t^2} - 2[y(t)]^2, \quad 0 \leqslant t \leqslant 10, \quad y(0) = 0.$$

using RK2 and RK4 formulas with stepsizes $h = 0.2, 0.1, 0.05, 0.025, 0.0125, 0.00625$. Plot the error functions in each case and determine numerical orders of convergence. The exact solution for this IVP is $y(t) = t/(1+t^2)$. Use this information to calculate errors and orders.

Lab Exercise 2.17. Use the RK2 method to solve

$$y'(t) = -y(t) + t^{0.1}(1.1 + t), \quad y(0) = 0$$

whose exact solution is $y(t) = t^{1.1}$. Solve the equation on interval $[0, 5]$ and compute the solution and errors at times $t = 1, 2, 3, 4, 5$. Use different stepsizes $h = 0.1, 0.05, 0.025, 0.0125, 0.00625$. Compute the order of convergence and compare with theoretical order 2 of the RK2 method. Explain your results.

Workout 2.18. What difficulty arises in attempting to use a Taylor series method of order $\geqslant 2$ to solve the equation

$$y'(t) = -y(t) + t^{0.1}(1.1 + t), \quad y(0) = 0.$$

What does it tell us about the solution?

Workout 2.19. (a) Write down the RK4 method for solving linear system of equations $\boldsymbol{y}'(t) = A\boldsymbol{y}(t)$ for $A \in \mathbb{R}^{n\times n}$ with initial condition $\boldsymbol{y}(0) = \boldsymbol{y}_0$. (b) How many matrix-vector multiplications should be performed in each step? (c) Optional: estimate the overall complexity of the method to approximate $\boldsymbol{y}(t_N)$.

3 Stiff differential equations

At the beginning of 1950's, a new difficulty was discovered in numerical solution of some practical ODEs, which has come to be known as **stiffness**, and led to some new concepts of stability by Germund Dahlquist[3] and others. Numerical methods with finite absolute stability regions (such as explicit methods) all fail to produce accurate and stable solutions for stiff problems unless the step size h is chosen excessively small which is impractical and inefficient in many situations.

3.1 What is stiffness?

It is difficult to formulate a precise definition for stiffness. One may argue that a stiff equation includes some terms that can lead to rapid variation (fast transients) in the solution. However, there exist some ODEs with smooth solutions[4] but are known as stiff problems. This means that the stiffness is independent of the solution but it is a property of the ODE itself. However, even if a stiff problem has a smooth solution, a slight perturbation to the solution at any time results in another solution curve that has a rapid variation. The following example from [**LeVeque:2007**] will make this more clear.

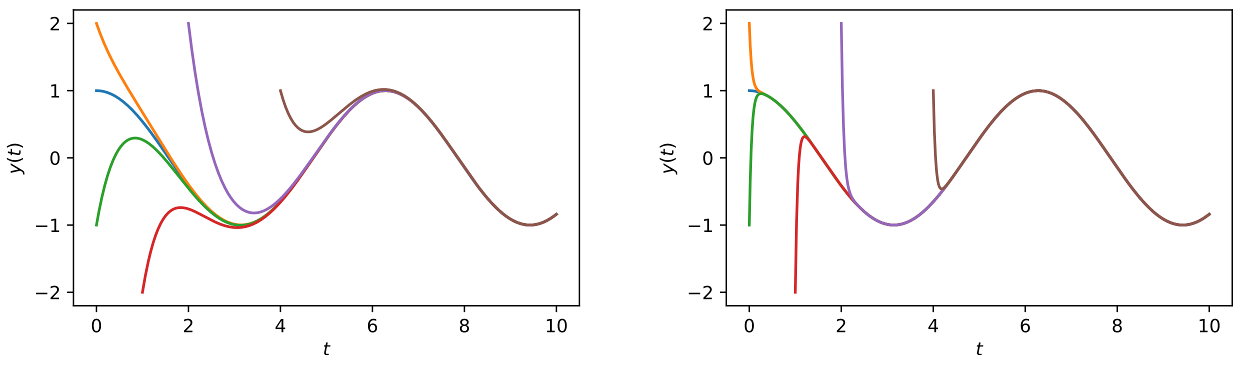
Example 3.1. Consider the ODE

$$y'(t) = \lambda(y - \cos t) - \sin(t) \tag{3.1}$$

from Example 2.4. For initial value $y(0) = 1$, this problem has smooth solution $y(t) = \cos t$ independent of the value of λ. If we change the initial condition to $y(t_0) = y_0$ that does not lie on this curve, then the solution is

$$y(t) = e^{\lambda(t-t_0)}(y_0 - \cos t_0) + \cos t$$

If $\mathrm{Re}(\lambda) < 0$, this function approaches $\cos t$ exponentially quickly with decay rate $\mathrm{Re}(\lambda)$. In Figure 13 different solution curves for this equation with different choices of t_0 and y_0 with $\lambda = -2$ and $\lambda = -20$ are plotted.



[3]Germund Dahlquist (1925–2005) was a Swedish mathematician known primarily for his early contributions to the theory of numerical solution of ODEs.

[4]In this section, by a 'smooth solution' we mean a function without any rapid transition.

Figure 13: Solution curves of a stiff problem with different initial times and initial values for $\lambda = -2$ (left) and $\lambda = -20$ (right).

We observe rapid transients in solution curves for $\lambda = -20$. The perturb solutions quickly approach toward the particular solution $y(t) = \cos t$. This problem is known as a stiff problem for values of λ with large real part magnitudes.

The phenomenon we observed in Example 3.1 will cause a serious numerical difficulty even if the initial condition is chosen such that the exact solution does not exhibit any rapid transient (for example $y(t) = \cos t$ with $y(0) = 1$ in ODE (3.1)). *Because any numerical method is subjected to local truncation and roundoff errors which act as a perturbation to the solution and move us away from the smooth solution to a solution with a rapid transient.* Numerical methods with finite absolute stability regions are unstable unless the time step is small relative to the time scale of the rapid transient. In the case of a smooth true solution it seems that a reasonably large step length would work, but the numerical method must always deal with the rapid transients introduced by truncation and roundoff errors in every time step. Consequently, a very small step length is needed to avoid the instability.

Example 3.2. Consider the ODE (3.1) on interval $[0, 10]$ with initial condition $y(0) = 1$. Let $\lambda = -10^4$. The numerical solution using the explicit RK4 method (2.37) with step length $h = 0.00028$ blows up as is shown in the left hand side of Figure 14. Smaller values of h such as $h = 0.00025$ leads to a stable solution that is shown in the right hand side. However, this stable calculations requires lots of function evaluations in the procedure of RK method. Note that, the complexity will quickly increase for a system of differential equations.

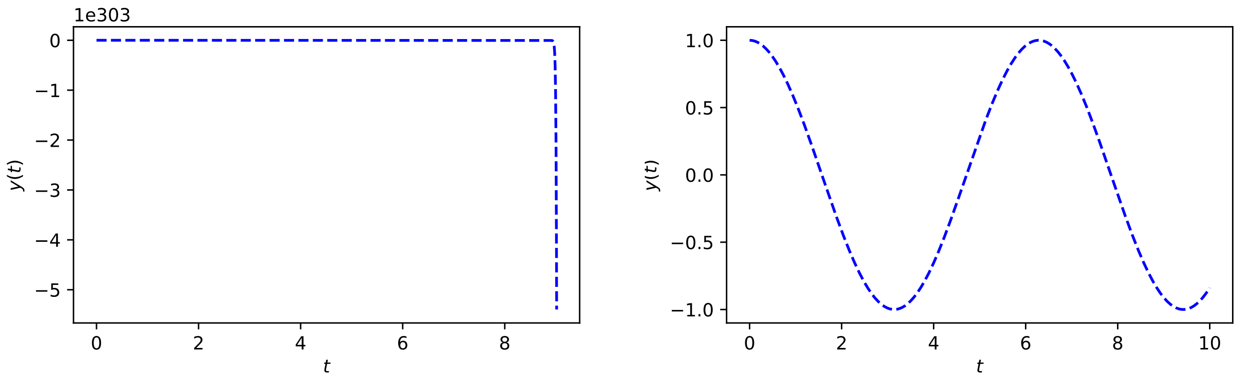


Figure 14: Numerical solution of ODE (3.1) with $\lambda = -10^4$ and initial condition $y(0) = 1$ using the RK4 method: unstable solution with $h = 0.00028$ (left) and a stable solution with $h = 0.00025$ (right). In the left panel, the values on the y-axis are multiplied by huge number 10^{303}.

It is better to look for other efficient numerical methods that can solve such a stiff problem stably using a much larger step size. For instance, the implicit Euler's method will do this job perfectly (write and execute the code). The trapezoidal method works much better than explicit methods but still introduces some limitations in the presence of rapid transients in the solution.

Lambert, after examining some other statements, finally has suggested the following definition for stiff problems [**Lambert:1991**].

Definition 3.1. A stiff ODE is an equation for which certain numerical methods with finite absolute stability regions for solving the equation are numerically unstable, unless the step size is taken to be excessively small in relation to the smoothness of the exact solution.

Explicit methods such as forward Euler's method and explicit RK methods (and in fact all explicit methods) are examples of numerical methods with finite absolute stability regions. Definition 3.1 reveals the fact that solving a stiff problem with an explicit method (or an implicit method with a finite absolute stability region) is very costly. We also note that the stiffness may vary over the total interval of integration.

3.2 Stiff systems

Now, let us look at a system of ODEs. Consider the linear system $\boldsymbol{y}'(t) = A\boldsymbol{y}(t) + \boldsymbol{g}(t)$ on interval $[0, b]$ with a constant matrix A of size $n \times n$, and initial condition $\boldsymbol{y}(0) = \boldsymbol{y}_0$. The exact solution of this ODE is

$$\boldsymbol{y}(t) = e^{At}\boldsymbol{y}_0 + \int_0^t e^{A(t-\tau)}\boldsymbol{g}(\tau)d\tau.$$

If A has distinct eigenvalues $\lambda_k \in \mathbb{C}$ and corresponding eigenvectors $\boldsymbol{v}_k \in \mathbb{C}^n$, then

$$\boldsymbol{y}(t) = \sum_{k=1}^{n} c_k e^{\lambda_k t}\boldsymbol{v}_k + \widetilde{\boldsymbol{g}}(t)$$

where $\boldsymbol{c} = V^{-1}\boldsymbol{y}_0$ for $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n]$, and $\widetilde{\boldsymbol{g}}(t)$ is the integral term (particular solution) in solution $\boldsymbol{y}(t)$. Let us assume that

$$\mathrm{Re}(\lambda_k) < 0, \quad k = 1, 2, \ldots, n,$$

which imply that all terms $e^{\lambda_k t}\boldsymbol{v}_k$ go to 0 as $t \to \infty$, so that the solution $\boldsymbol{y}$ approaches $\widetilde{\boldsymbol{g}}(t)$ asymptotically as $t \to \infty$. The term $e^{\lambda_k t}\boldsymbol{v}_k$ decreases monotonically if λ_k is real and sinusoidally if λ_k is complex. Thus, the term

$$\sum_{k=1}^{n} c_k e^{\lambda_k t}\boldsymbol{v}_k$$

can be viewed as the *transient solution* and the term $\widetilde{\boldsymbol{g}}(t)$ as the *steady state* solution. If $\mathrm{Re}(\lambda_k)$ is large then the corresponding term $c_k e^{\lambda k t}\boldsymbol{v}_k$ will decay quickly as t increases and is thus called a fast transient; if $\mathrm{Re}(\lambda_k)$ is small the corresponding term $c_k e^{\lambda k t}\boldsymbol{v}_k$ decays slowly and is called a slow transient. Let $\overline{\lambda}$ and $\underline{\lambda}$ be defined such that

$$|\,\mathrm{Re}(\underline{\lambda})| \leqslant |\,\mathrm{Re}(\lambda_k)| \leqslant |\,\mathrm{Re}(\overline{\lambda})|, \quad k = 1, 2, \ldots n.$$

If our aim is to reach the steady-state solution, then we must keep integrating until the slowest transient is negligible. The smaller $|\,\mathrm{Re}(\underline{\lambda})|$ is, the longer we must keep integrating. If, however, the method we are using has a finite region of absolute stability, we must ensure that the step size h is sufficiently small for $\lambda_k h \in S$, $k = 1, 2, \ldots, n$ to hold. Clearly a large value of $|\operatorname{Re}(\overline{\lambda})|$ implies a small step size h. We therefore get into a difficult situation if $|\operatorname{Re}(\overline{\lambda})|$ is very large and $|\operatorname{Re}(\underline{\lambda})|$ is very small; we are forced to integrate for a very long time with an excessively small step size. It seems natural to take the ratio

$$r_S := \frac{|\operatorname{Re}(\overline{\lambda})|}{|\operatorname{Re}(\underline{\lambda})|}$$

the *stiffness ratio*, as a measure of the stiffness of the system [**Lambert:1991**]. While r_S is often a useful quantity, one should not rely entirely on this measure to determine whether a problem is stiff. For example, a scalar problem can be stiff while the r_S is always 1 since there is only one eigenvalue. Or, in a system of equations if one eigenvalue is zero then the contribution of this eigenvalue to the exact solution is a constant term. If the moduli of the real parts of the remaining eigenvalues are not particularly large, the system is not stiff, yet the stiffness ratio is now infinite.

Example 3.3. Consider the following systems of ODEs [**Lambert:1991**]

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} 2\sin t \\ 2(\cos t - \sin t) \end{bmatrix}, \quad \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix} \tag{3.2}$$

and

$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ 998 & -999 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} 2\sin t \\ 999(\cos t - \sin t) \end{bmatrix}, \quad \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}. \tag{3.3}$$

Both problems have the same exact solution

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = 2e^{-x} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} \sin t \\ \cos t \end{bmatrix}.$$

The plots of these solutions on interval $[0, 10]$ are given in Figure 15. Both solutions are smooth (without any rapid transients).

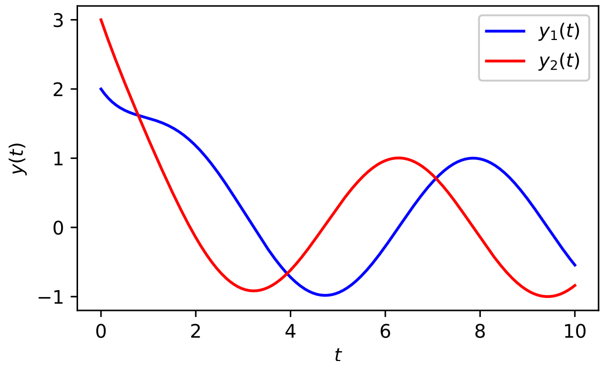


Figure 15: Exact solutions of systems (3.2) and (3.3)

We employ the explicit Euler's method for both systems. Everything is perfect for system (3.2), but unstable results are obtained for system (3.3) unless the step size is chosen smaller than 0.002. System (3.3) is stiff but system (3.2) is non-stiff. The eigenvalues of the matrix in (3.2) are -1 and -3, and if we consider the general initial condition to $y(0) = y_0$ then the exact solution is

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = c_1 e^{-t} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + c_2 e^{-3t} \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} \sin t \\ \cos t \end{bmatrix}.$$

where c_1 and c_2 are determined by imposing the initial value y_0. The eigenvalues of the matrix in (3.3) are -1000 and -1 and the exact solution for an arbitrary initial value y_0 is

$$\begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = c_1 e^{-t} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + c_2 e^{-1000t} \begin{bmatrix} 1 \\ -998 \end{bmatrix} + \begin{bmatrix} \sin t \\ \cos t \end{bmatrix}.$$

In the second solution the exponential term e^{-1000t} produces a rapid transient in the solution. Although the initial value $y_0 = [2, 3]^T$ gives $c_1 = 2$ and $c_2 = 0$ for both systems (therefore annihilates the rapid transient in the second system), slight perturbations (truncation and roundoff errors) in numerical solution at different time levels produce a component on the rapid transient term and introduce the mentioned numerical difficulties. The same happens even if we choose initial conditions such that $c_1 = c_2 = 0$; in that case the explicit Euler's and explicit RK methods are unable to integrate even the very smooth solution $y(x) = [\sin t, \cos t]^T$ unless at very small stepsizes. Finally, we note that the stiffness ratio is $r_S = 3$ for system (3.2) while it is $r_S = 1000$ for system (3.3).

The situation would be more complicated for nonlinear system of equations. For the scaler case the partial derivative $\frac{\partial f}{\partial y}$ or Lipschitz constant of f with respect to y, and for nonlinear system the eigenvalues of the Jacobian matrix $J_f(t, \boldsymbol{y})$ may give an insight to discover the stiffness. As we pointed out at the end of subsection 1.4, the stability analysis through the Jacobian matrix has only a local validity. Nevertheless, the stiffness ratio

$$r_S = \max_{t\in[t_0,b]} \frac{\max_k |\operatorname{Re}(\lambda_k(t))|}{\min_k |\operatorname{Re}(\lambda_k(t))|},$$

where $\lambda_k(t)$ are eigenvalues of $J_f(t, \boldsymbol{y})$, may give an insight to stiffness of system $\boldsymbol{y}'(t) = f(t, \boldsymbol{y})$.

Remark 3.1. Sometimes we can indicate the stiffness of system of ODEs $\boldsymbol{y}'(t) = f(t, \boldsymbol{y})$ by looking at the coefficients on the right-hand side. When these coefficients vary significantly in magnitude, it suggests that the system is likely stiff. For example the following system of equations for Robertson's auto-catalytic chemical reaction

$$\begin{aligned} y_1' &= -0.04y_1 + 10^4 y_2 y_3 \\ y_2' &= 0.04y_1 - 10^4 y_2 y_3 - 3 \times 10^7 y_2^2 \\ y_3' &= 3 \times 10^7 y_2^2 \end{aligned}$$

is a stiff system of ODEs, as it contains coefficients with different sizes ranging from 0.04 to 3×10^7. This is also the case for system (3.3) in Example 3.3.

A practical way to detect the stiffness of an ODE is to attempt to solve it using a method with a finite absolute stability region with a moderate step size and see whether the computed solution is blown up or is not.

Workout 3.2. Consider solving Robertson's auto-catalytic chemical system of ODEs, as described in Remark 3.1, over the time interval $[0, 500]$ with initial conditions $y_1(0) = 1$, $y_2(0) = y_3(0) = 0$, using the Explicit Euler and Explicit RK4 methods. Ensure that the chosen step length h is sufficiently small to yield stable results. Upon plotting the solutions, provide your observations. What distinguishes the behavior of the solution y_2 from the others?

3.3 A-stability

It is clear from the considerations of this section that those methods with bounded stability regions are inappropriate for stiff problems. Such class of methods includes all explicit methods and even some implicit methods such as Adams-Moulton methods. On the other hand, an implicit method that its region of absolute stability includes the whole of the left half-plane is efficient for stiff problems because there will be no stability restriction on the step size h provided that all the eigenvalues have negative real parts, as is often the case in practice.

Definition 3.3. A numerical ODE solver is called **A-stable** if its absolute stability region contains the whole left-half plane $\{z \in \mathbb{C} : \text{Re}(z) \leqslant 0\}$.

Among the methods we have already studied, the implicit Euler's method and the trapezoidal method are A-stable. As a negative result, the *Dahlquist's second barrier theorem* states that any A-stable linear multistep method is at most second order accurate, and in fact the trapezoidal method is the A-stable method with smallest truncation error. However, higher order A-stable implicit RK methods (multi-stage methods) do exist.

For many stiff problems the eigenvalues are located near the negative real axis or at most in a wedge $\pi - \alpha \leqslant \arg(z) \leqslant \pi + \alpha$ for an angle $\alpha \in [0, \pi/2]$. For such problems, we just need the stability region to contain such a wedge rather than the whole left-half plane. See Figure 16.

Definition 3.4. A numerical ODE solver is called **A(α)-stable**, for $\alpha \in [0, \pi/2]$, if its absolute stability region contains the wedge $\{z \in \mathbb{C} : \pi - \alpha \leqslant \arg(z) \leqslant \pi + \alpha\}$.

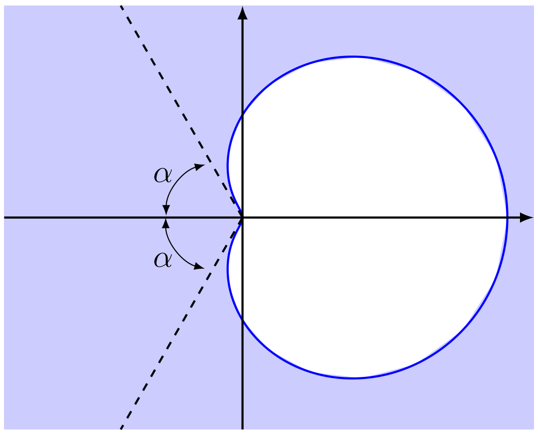


Figure 16: A region of A(α)-stability

An A-stable method is A($\pi/2$)-stable. A method is A(0)-stable if the negative real axis itself lies in the stability region. Some numerical methods with A(α)-stability property which are appropriate for stiff ODE problems have been developed. For example we can mention the **implicit RK** methods and the **backward differentiation formulas (BDF)** which will be introduced in sections 5 and 6.3. To see how to solve stiff ODEs using MATLAB, refer to the section 7 below.

3.4 L-stability

As we pointed out, both implicit Euler and trapezoidal methods are A-stable but there is a major difference between the stability regions of these methods. The trapezoidal method is stable only in the left half-plane, whereas implicit Euler's method is stable not only in the left-half plane but also over much of the right half-plane. See Figures 9 and 11. Let us solve the stiff ODE (3.1) on interval $[0, 10]$ for $\lambda = -10^4$ with both methods. First, we assume that $y(0) = 1$ which corresponds to the smooth exact solution $y(t) = \cos t$. Let $h = 0.2$. Both methods provide satisfactory results with norm-infinity error 9.998×10^{-6} for the Euler's method and 3.346×10^{-7} for the trapezoidal method. Remember that the explicit methods failed to numerically solve this problem even at much smaller step sizes. It is expectable that the trapezoidal method is more accurate because its convergence order is 2 compared with the convergence order of the implicit Euler's method which is only 1. However, this is not the whole story. Let us change the initial value to $y(0) = 1.5$ which corresponds to exact solution

$$y(t) = \frac{1}{2}e^{-10000t} + \cos t$$

which includes a fast transient term. The initial value $y(0) = 1.5$ rapidly (at a time scale of about 10^4) decreases toward the particular solution $\cos t$. The approximate solutions with $h = 0.2$ are plotted in Figure 17.

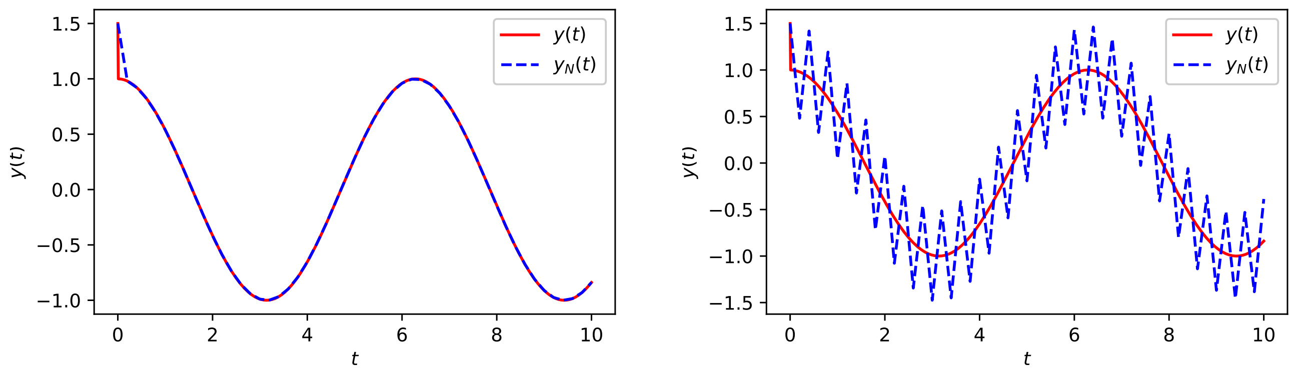


Figure 17: Numerical solution of stiff problem (3.1) with $\lambda = -10^4$ and $y(0) = 1.5$ with step size $h = 0.2$, the implicit Euler's method (left) and the trapezoidal method (right).

Both methods are still absolutely stable, but the result of trapezoidal method shows unsatisfactory oscillations. For absolute stability we test on equation $y' = \lambda y$ and obtain $y_{k+1} = R(z)y_k$

such that

$$R(z) = \frac{1}{1-z}, \quad |R(z)| \to 0 \text{ as } z \to \infty,$$

for the implicit Euler's method and

$$R(z) = \frac{1+\frac{1}{2}z}{1-\frac{1}{2}z}, \quad |R(z)| \to 1 \text{ as } z \to \infty,$$

for the trapezoidal method. For problems with rapid transients, we aim for a method that can effectively damp in a single time step, because we intend to use a steplength much larger than the true decay time of the transient. To illustrate this point, refer to Figure 18, which provides a close-up comparison of the exact and numerical solutions obtained with each method over the time interval $[0, 2]$.

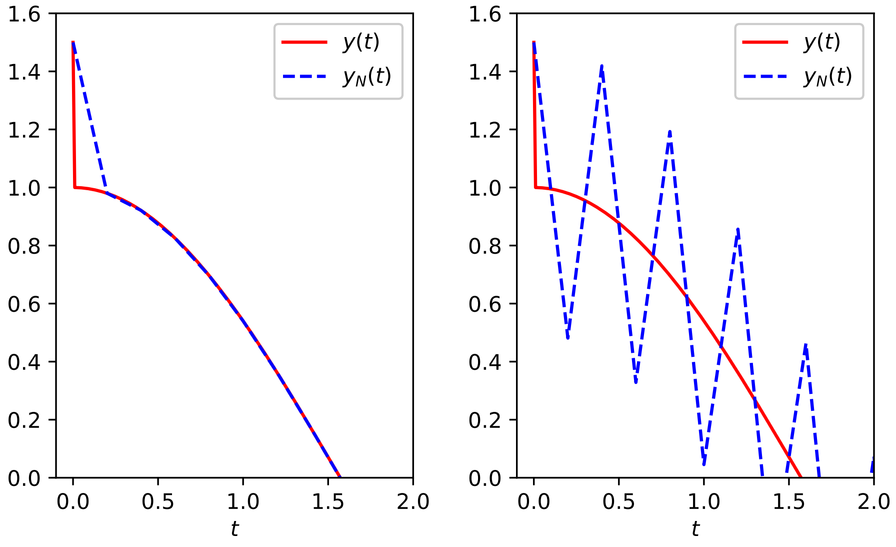


Figure 18: Closeup of solutions: the implicit Euler's method (left) and the trapezoidal method (right).

At the first time step, the implicit Euler's method damps very effectively toward the steady state solution $\cos t$ and continues to produce very accurate results thereafter. In fact, this method, if is applied on stiff ODE (3.1), yields

$$y_{k+1} = \frac{1}{1+\lambda h} y_k - \frac{\lambda h}{1+\lambda h} \cos t_{k+1} - \frac{h}{1-\lambda h} \sin t_{k+1},$$

which means

$$y_{k+1} \approx \cos t_{k+1}$$

because $\lambda h = -10^4 \times 0.2 = -2000$ and

$$\frac{1}{1+\lambda h} = \frac{1}{2001} \doteq 0.0005 \approx 0.$$

This implies that by the second time step we approximately fall on the steady state solution $\cos t$. The trapezoidal method is also stable and the results stay bounded, however, we have

$$\frac{1+\frac{1}{2}\lambda h}{1-\frac{1}{2}\lambda h} = -\frac{999}{1001} \doteq -0.9980 \approx -1.$$

By applying on stiff ODE (3.1), this method gives

$$y_{k+1} = \frac{1+\frac{1}{2}\lambda h}{1-\frac{1}{2}\lambda h} y_k - \frac{\frac{1}{2}\lambda h}{1-\frac{1}{2}\lambda h}[\cos t_{k+1} + \cos t_k] - \frac{\frac{1}{2}h}{1-\frac{1}{2}\lambda h}[\sin t_{k+1} + \sin t_k],$$

or,

$$y_{k+1} \approx -y_k + [\cos t_{k+1} + \cos t_k].$$

At the first time step, we have $y_1 \approx -1.5 + [\cos 0 + \cos 0.2] \approx 0.48$ which falls below the steady state solution. Moving to the second time step, y_2 is $0.48 + [\cos 0.2 + \cos 0.4] \approx 2.38$, overshooting the steady-state solution. This pattern persists in subsequent time steps, resulting in the zigzag-shaped solution observed on the left-hand side of Figure 17.

The results of the trapezoidal method can be improved if a small enough steplength h is used. In Figure 19, numerical solutions with $h = 10^{-2}$ and $h = 10^{-4}$ are shown.

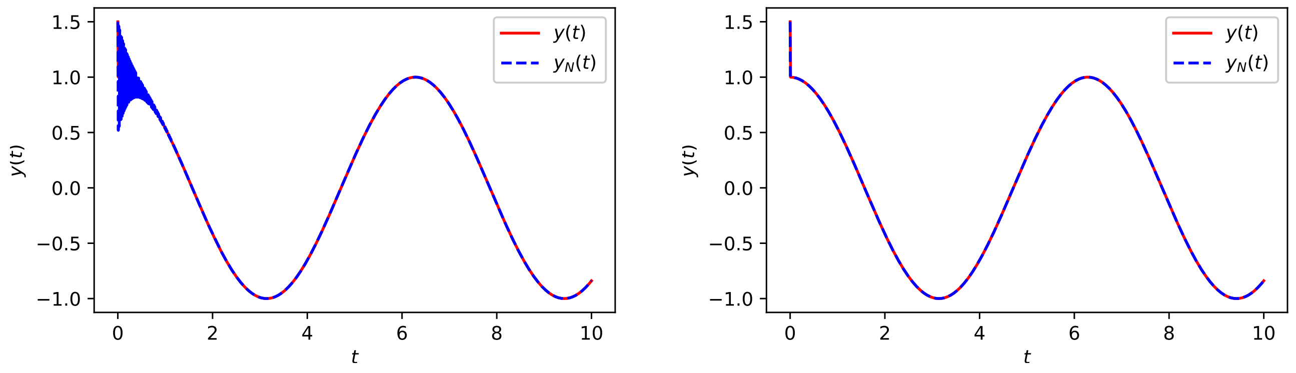


Figure 19: Numerical solution of stiff problem (3.1) with $\lambda = -10^4$ and $y(0) = 1.5$ using the trapezoidal method, with $h = 10^{-2}$ (left) and $h = 10^{-4}$ (right).

With step length $h = 10^{-2}$ we still have oscillations near the initial time, while with step length $h = 10^{-4}$ a perfect numerical solution is observed. We note that, $h = 10^{-4}$ is identical with the time scale of the transient term. This means that the trapezoidal method works perfect provided that the step size is chosen equal or smaller than the time scale of the transient terms in the solution.

However, our primary aim was to develop efficient numerical methods that be able to integrate the stiff problems with a rather large step length. What which makes the implicit Euler's method different from the trapezoidal method is the following property that the implicit Euler's method possesses while the trapezoidal method does not.

Definition 3.5. A one-step method is called **L-stable** if it is A-stable and $\lim_{z\to\infty} |R(z)| = 0$.

The implicit Euler method is L-stable. We will introduce some higher order L-stable methods in the forthcoming sections.

4 Adaptive time stepping

All solvers presented up to here use a single stepsize h in all iterations. It is desirable to have an algorithm that can adjust the stepsize h_k in each step k to ensure that the local truncation errors at all steps remain below a certain tolerance:

$$|\tau_k| \leqslant \varepsilon, \text{ for all } \ k. \tag{4.1}$$

In certain parts of the domain, where the derivatives of y have small amplitudes, larger stepsizes may suffice, resulting in a reduced computational expense due to a lower number of function evaluations. Since y and its derivatives are unknown the truncation error can not be calculated analytically. To do so, we need an **estimate** for τ_k at each step, and a way to select a new stepsize that will ensure that the estimated error is acceptably small.

Remark 4.1. It is more reasonable to keep the **global error** under control rather than the **local errors** (4.1). Because bounding the τ_k's individually does not lead to a natural bound on the global error, since it ignores the propagation of the error in each step. But local control is simple and is good enough in practice if we take local tolerances smaller than what it seems necessary. See the following.

Now we address the question of how to estimate the local errors and compute the new time step.

4.1 Using two methods of different order

A good strategy is to use two methods of different orders. Informally, we use the more accurate one as an estimate for the exact solution, to estimate the error for the less accurate method. Assume that for a given stepsize h we have two methods:

- Method A: with local order p and truncation error $\tau_{k+1}(h)$ to compute y_{k+1},
- Method B: with local order $p+1$ to compute a more accurate value $\tilde{y}_{k+1}$.

Then the estimated local truncation error is

$$e_{k+1} := \tilde{y}_{k+1} - y_{k+1}.$$

If we suppose that the value at time step k is exact, i.e. $y_k = y(t_k)$, then by definition of local truncation errors we have

$$\begin{aligned} y(t_{k+1}) &= y_{k+1} + \tau_{k+1}(h) \\ y(t_{k+1}) &= \tilde{y}_{k+1} + \mathcal{O}(h^{p+1}). \end{aligned} \tag{4.2}$$

Since the local error of Methpd A is $\mathcal{O}(h^p)$, we can expand the truncation error $\tau_{k+1}(h)$ as

$$\tau_{k+1}(h) = Ch^p + \mathcal{O}(h^{p+1}),$$

for a constant C. By subtracting two equations in (4.2) and taking absolute value we obtain

$$|e_{k+1}| = |\tilde{y}_{k+1} - y_{k+1}| = |C|h^p + \mathcal{O}(h^{p+1}).$$

By dropping the term $\mathcal{O}(h^{p+1})$, we approximately have

$$|\tau_{k+1}(h)| \approx |C|h^p \approx |e_{k+1}|. \tag{4.3}$$

Consequently, if the error is small, i.e.,

$$|e_{k+1}| \leqslant \varepsilon \tag{4.4}$$

then the step is *accepted* and the more accurate solution $\tilde{y}_{k+1}$ is assigned as an approximation for $y(t_{k+1})$. The algorithm then goes to the next time step $t_{k+1} + h$. Otherwise the tentative value $\tilde{y}_{k+1}$ is *rejected* and the step must be redone by a smaller steplength h_{new}. To estimate h_{new} we use this fact that it must satisfy

$$|\tau_{k+1}(h_{new})| \approx |C|h_{new}^p \leqslant \varepsilon.$$

Taking the ratio of this and the estimate (4.3) we obtain

$$\left(\frac{h_{new}}{h}\right)^p \lesssim \frac{\varepsilon}{|e_{k+1}|}$$

which gives an estimate for h_{new} as

$$h_{new} \lesssim h\left(\frac{\varepsilon}{|e_{k+1}|}\right)^{1/p}.$$

In practice, because this is just an estimate, one puts an extra coefficient in to be safe, typically something like

$$h_{new} := 0.8h\left(\frac{\varepsilon}{|e_{k+1}|}\right)^{1/p}. \tag{4.5}$$

This formula decreases the stepsize if the error is large. This process should be repeated by replacing h by h_{new} until (4.4) is satisfied and the step is accepted. Sometimes, other controls are added; like not decreasing or increasing h by too much per time step.

4.2 Embedded RK methods

A disadvantage of the above adaptive scheme is that each step involving error estimation incurs a cost that is roughly twice that of a fixed step method because two methods are run. Nonetheless, by carefully selecting the methods, we can generate some computational overlap and thereby reduce the workload.

An approach is to use an **embedded pair** of RK formulas where most of the f values are the same for both methods A and B. For example, suppose we wish to create a pair for estimating the error in Euler's method (RK1) as Method A. We also need a method with order 2 (local order 3) as Method B. Recall the RK2 method (2.32) which can be written as

$$\begin{aligned} f_1 &= f(t_k, y_k) \\ f_2 &= f\Big(t_k + \frac{h}{2}, y_k + \frac{h}{2}f_1\Big) \\ \tilde{y}_{k+1} &= y_k + hf_2 \end{aligned}$$

Then the value y_{k+1} is computed via the Euler's method by

$$y_{k+1} = y_k + hf_1,$$

which requires essentially no extra function evaluation; the value of f_1 is used for both. Then following the rule (4.5) the adapted stepsize is computed as

$$h_{new} = 0.8h \left(\frac{\varepsilon}{|e_{k+1}|} \right)^{1/2}$$

where $e_{k+1} = \tilde{y}_{k+1} - y_{k+1} = h(f_2 - f_1)$. That value y_{k+1} is not actually needed to compute because the error estimation e_{k+1} is obtained in terms of f_k values, and the more accurate value $\tilde{y}_{k+1}$ is selected as an approximation for $y(t_{k+1})$. The initial stepsize $h = \varepsilon^{1/3}$ can be used because the local error in the first step is supposed to be of order h^3. The MATLAB code for such scheme is given here.

```matlab
function [T,Y] = ODE12(f,y0,tspan,tol)
% Adaptive embedded method with RK1 and a RK2 formulas
% Inputs:
%  f: right hand side function f(t,y)
%  y0: initial condition
%  tspan: [t0, tfinal]
%  tol: predefined error
% Output:
%  T: vector of time steps
%  Y: solution
t = tspan(1); Y = y0; T = t;
h = tol^(1/3);
while t <= tspan(2)
   f1 = f(t,y0);
   f2 = f(t+h/2,y0+h/2*f1);
   e = norm(h*(f2-f1),inf);
   if e < tol
      yt = y0 + h*f2;
      y0 = yt; t = t + h;
      Y = [Y yt]; T = [T t];
      h = tol^(1/3);
   else
      h = 0.8*h*(tol/e)^(1/2);
   end
end
```

Example 4.1. As an example, consider the IVP

$$y' = \frac{1}{y^2 + 0.01}, \quad 0 \leqslant t \leqslant 3,$$
$$y(0) = 0$$

We solve this equation using the above MATLAB code with tolerance $\varepsilon = 10^{-3}$.

```
[t,y] = ODE12(@(t,y) 1/(y^2+0.01),0,[0 3],10e-3);
plot(t,y(1,:),'-ob','MarkerFaceColor','b')
```

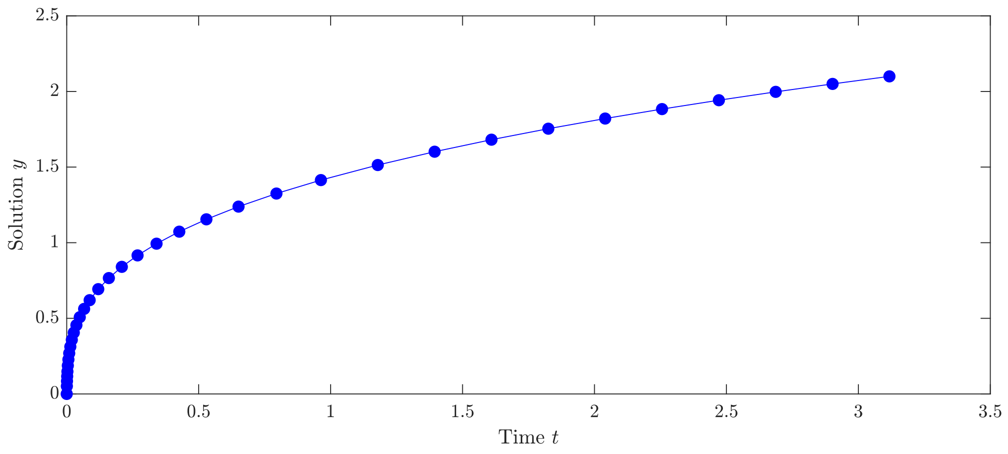


Figure 20: Numerical solution using the adaptive scheme

Plot of the solution is given in Figure 20. The scheme adapts much smaller stepsizes at the beginning of the time interval (where the solution takes more action) but remarkably larger stepsizes at remaining parts. This leads to saving memory and time specially when the underlying problem is a large system of equations with a long time interval.

Embedded methods of higher order can be constructed by the right choice of coefficients. One popular embedded pair is the **Runge-Kutta-Fehlberg** method, which uses a fourth-order and fifth-order formula that share most of the f_k values. A formula of this form with stepsize selected by (4.5) is the strategy employed, for instance, by MATLAB's `ode45`.

5 Implicit Runge-Kutta methods

While high convergence order and ease of implementation are advantages of explicit RK methods and make them popular for solving various types of ODEs, their bounded stability regions render them impractical for a variety of important and challenging problems, such as stiff differential equations. Therefore, it is natural to develop, among other techniques, a class of implicit RK methods.

An s-stage **implicit Runge-Kutta** method has the form

$$
\begin{aligned}
z_\ell &= y_k + h\sum_{j=1}^{s} a_{\ell j} f(t_k + c_j h, z_j), \quad \ell = 1,2,\dots,s,\\
y_{k+1} &= y_k + h\sum_{j=1}^{s} b_j f(t_k + c_j h, z_j).
\end{aligned}
$$

The Butcher's tableau for this formula has the form

$$
\begin{array}{c|cccc}
c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,s}\\
c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,s}\\
\vdots & \vdots & \vdots & & \vdots\\
c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s}\\
\hline
 & b_1 & b_2 & \cdots & b_s
\end{array}
$$

Here, the diagonal and upper diagonal parts of the tableau may have nonzero values. To advance form time level t_k to t_{k+1} using an s-stage implicit RK method we should solve a system of s nonlinear equations for s unknowns $z_1, z_2, \dots, z_s$. For a system of equations $\boldsymbol{y}' = f(t, \boldsymbol{y})$ with m differential equations we must solve a system of sm equations in sm unknowns at each time step.

There typically are some ways to derive coefficients $\{b_j, c_\ell, a_{\ell,j}\}$ for a given accuracy, provided the number of stages is sufficiently large. The dominant approach converts the IVP in to integral equation

$$
y(t) = y(t_k) + \int_{t_k}^{t} f(\tau, y(\tau))d\tau, \quad t \in [t_k, t_{k+1}],
$$

uses a polynomial interpolation of order s for $f(\tau, y(\tau))$ on a predefined set of interpolation points $\tau_1, \dots, \tau_s \in [t_k, t_{k+1}]$, and collocate the resulting equation at $t = \tau_1, \dots, \tau_s$ to predict $z_1, z_2, \dots, z_s$ and y_{k+1}. This approach is called *collocation.* We omit derivation details but we present the Butcher's tableaus for some formulas instead. An extensive theory has been developed by Butcher for analyzing these methods.

The tableau for a two-stage formula with good convergence and stability properties is

$$
\begin{array}{c|cc}
(3-\sqrt{3})/6 & 1/4 & (3-2\sqrt{3})/12\\
(3+\sqrt{3})/6 & (3+2\sqrt{3})/12 & 1/4\\
\hline
 & 1/2 & 1/2
\end{array}
$$

This method is also called *two-stage Gauss method* because the transferred Gauss-Legendre points $\{-\frac{\sqrt{3}}{3}, \frac{\sqrt{3}}{3}\}$ are used for the polynomial interpolation. It can be shown that the local truncation error for this method has size $\mathcal{O}(h^5)$, and the global error behaves like $\mathcal{O}(h^4)$.

Workout 5.1. Show that the absolute stability region of the two-stage Gauss method is

$$
S = \left\{ z \in \mathbb{C} : \left| \frac{1 + \frac{1}{2}z + \frac{1}{12}z^2}{1 - \frac{1}{2}z + \frac{1}{12}z^2} \right| \leqslant 1 \right\}.
$$

Show that the negative part of the real line falls in S. More generally, show that the left-half plane falls in S.

The nice stability feature of the two-stage Gauss method, as outlined in Workout 5.1, comes at the cost of solving a nonlinear system of algebraic equations for each time step. In general, a fully implicit RK method where each z_ℓ value depends on all the z_ℓ values, can be costly to implement for a system of ODEs. This is because a nonlinear system of sm equations in sm unknowns must be solved at each time step. However, a subclass of implicit methods that are simpler to implement are the **diagonally implicit Runge–Kutta** methods (DIRK methods), in which $a_{\ell,j} = 0$ for $j > \ell$, i.e., z_ℓ depends on z_j for $j = 1, \ldots, \ell$. An s-stage DIRK method, when applied to a system of m nonlinear ODEs, requires solving a sequence of s nonlinear systems, each of size m, rather than a coupled set of sm equations.

As an example of a 3-stage DIRK method, we can mention a method with following Butcher tableau:

$$\begin{array}{r|ccc} 0 & 0 & & \\ 1/2 & 1/4 & 1/4 & \\ 1 & 1/3 & 1/3 & 1/3 \\ \hline & 1/3 & 1/3 & 1/3 \end{array}$$

This method is of second order accuracy, and also is known as the TR-BDF2 method.

6 Multistep methods

All methods we considered so far are *single-step* or *one-step* methods, since at a typical step y_{k+1} is determined solely from y_k. In this section we study the *multistep methods* where the solution y_{k+1} depends on more than one previous values, i.e., $y_k, y_{k-1}, \ldots$. Three families of such methods are widely used; **Adams-Bashforth (AB)** and **Adams-Moulton (AM)** methods and **backward differentiation formulas (BDF)**.

Again, we reformulate the solution of IVP $y'(t) = f(t, y)$ at $t = t_{k+1}$ as

$$y(t_{k+1}) = y(t_k) + \int_{t_k}^{t_{k+1}} f(t, y(t))dt. \tag{6.1}$$

A numerical scheme for computing $y(t_{k+1})$ can be obtained if the integral on the right-hand side of (6.1) is approximated by a numerical quadrature. A numerical quadratures for computing a definite integral of the form

$$\int_{t_k}^{t_{k+1}} g(t)dt$$

can be developed by replacing g with an interpolation polynomial p of a certain degree. In a q-step AB method we assume that p interpolates g at points $\{t_{k-q}, \ldots, t_{k-1}, t_k\}$ while in a q-step AM method at points $\{t_{k-q+1}, \ldots, t_k, t_{k+1}\}$. In AM methods the solution y_{k+1} depends on already computed values $y_k, y_{k-1}, \ldots, y_{k-q}$ thus AB methods are explicit. In AM methods, on the other hand, y_{k+1} depends on $q - 1$ previous values and y_{k+1} itself, thus AM methods are implicit.

6.1 Adams-Bashforth methods

Let $q = 1$. The linear interpolant of g at points $\{t_{k-1}, t_k\}$ then is

$$p_1(t) = \frac{1}{h}[(t_k - t)g(t_{k-1}) + (t - t_{k-1})g(t_k)],$$

with error function

$$e(t) = g(t) - p_1(t) = \frac{1}{2}(t_k - t)(t - t_{k-1})g''(\zeta_k),$$

for some $t_{k-1} \leqslant \zeta_k \leqslant t_k$. Integrating over $[t_k, t_{k+1}]$ yields

$$\int_{t_k}^{t_{k+1}} g(t)dt = \int_{t_k}^{t_{k+1}} p_1(t) + \int_{t_k}^{t_{k+1}} e(t) = \frac{h}{2}[3g(t_k) - g(t_{k-1})] + \frac{5h^3}{12}g''(\xi_k)$$

for some $t_{k-1} \leqslant \xi_k \leqslant t_{k+1}$. Applying this to (6.1), gives

$$y(t_{k+1}) = y(t_k) + \frac{h}{2}\Big[3f(t_k, y(t_k)) - f(t_{k-1}, y(t_{k-1}))\Big] + \frac{5h^3}{12}y'''(\xi_k). \qquad (6.2)$$

Dropping the error term and replacing the exact values $y(t_k)$ by the approximate values y_k, we obtain the 2-step AB formula

$$y_{k+1} = y_k + \frac{h}{2}\Big[3f(t_k, y_k) - f(t_{k-1}, y_{k-1})\Big], \quad k = 1, 2, \ldots. \qquad (6.3)$$

At the initial level $k = 1$, computing y_2 requires both y_0 and y_1, yet we only have y_0 from the initial value. The value of y_1 must be computed using another method. If y_1 is obtained in such a way that $|y_1 - y(t_1)| \leqslant c_1 h^2$ then it can be proved that the global error of the method (6.3) satisfies

$$|e_N| \leqslant Ch^2$$

provided that h is sufficiently small, $f(t, y)$ is Lipschitz continuous and y is 3 times continuously differentiable. See section 6.4. To calculate y_1 we have at least two possibilities from previous sections. The explicit Euler's method gives

$$y_1 = y_0 + hf(t_0, y_0)$$

with error $|y(t_1) - y_1| \leqslant c_1 h^2$, and the RK2 method

$$y_1 = y_0 + \frac{h}{2}[f(t_0, y_0) + f(t_0 + h, y_0 + hf(t_0, y_0))]$$

with error $|y(t_1) - y_1| \leqslant c_1 h^3$, which is more than adequate.

The 3-step AB method is obtained by interpolating g at points $\{t_{k-2}, t_{k-1}, t_k\}$ and then integrating over $[t_k, t_{k-1}]$ as is required in (6.1). The interpolant will be

$$p_2(t) = \ell_0(t)g(t_k) + \ell_1(t)g(t_{k-1}) + \ell_2(t)g(t_{k-2})$$

with Lagrange functions

$$\ell_0(t) = \frac{1}{2h^2}(t - t_{k-1})(t - t_{k-2}),$$
$$\ell_1(t) = \frac{1}{h^2}(t - t_k)(t - t_{k-2}),$$
$$\ell_2(t) = \frac{1}{2h^2}(t - t_k)(t - t_{k-1}),$$

and interpolation error

$$e(t) = g(t) - p_2(t) = \frac{1}{6}(t - t_k)(t - t_{k-1})(t - t_{k-2})g'''(\zeta_k)$$

for some $t_{k-2} \leqslant \zeta_k \leqslant t_k$. Exact integration of the polynomial p_2 and error term e reveals that

$$\int_{t_k}^{t_{k+1}} g(t)dt = \frac{h}{12}\left[23g(t_k) - 16g(t_{k-1}) + 5g(t_{k-1})\right] + \frac{3h^4}{8}g'''(\xi_k)$$

for some $t_{k-2} \leqslant \xi_k \leqslant t_{k+1}$. Using this quadrature for (6.1) and dropping the error term, we obtain the 3-step AB method

$$y_{k+1} = y_k + \frac{h}{12}\left[23y'_k - 16y'_{k-1} + 5y'_{k-2}\right], \quad k = 2, 3, 4, \dots, \tag{6.4}$$

where $y'_k = f(t_k, y_k)$. In this formula y_1 and y_2 values must be obtained separately by other methods with errors at most of order h^3 (such as RK2 method) to keep the global error of (6.4) of order h^3.

Higher order AB methods can be derived similarly. In Table 4 the AB methods of order 1 through 4 are listed. Local truncations errors are given in the last column. See section 6.4 for an error analysis.

Table 4: Adams-Bashfoth methods. Here by y'_k we mean $f(t_k, y_k)$.

q	Order	Method	τ_{k+1}
0	1	$y_{k+1} = y_k + hy'_k$	$\frac{1}{2}h^2y''(\xi_k)$
1	2	$y_{k+1} = y_k + \frac{1}{2}h[3y'_k - y'_{k-1}]$	$\frac{5}{12}h^3y'''(\xi_k)$
2	3	$y_{k+1} = y_k + \frac{1}{12}h[23y'_k - 16y'_{k-1} + 5y'_{k-2}]$	$\frac{3}{8}h^4y^{(4)}(\xi_k)$
3	4	$y_{k+1} = y_k + \frac{1}{24}h[55y'_k - 59y'_{k-1} + 37y'_{k-2} - 9y'_{k-3}]$	$\frac{251}{720}h^5y^{(5)}(\xi_k)$

Compared to Runge-Kutta methods with the same order of accuracy, multistep methods require fewer evaluations of f at each time step. For instance, in the explicit RK4 method (2.37), we need 4 function evaluations per time step, whereas in the explicit 4-step AB method, only 1 function evaluation is needed in each time step, provided that previous function values of f are reused.

6.2 Adams-Moulton methods

The implicit Euler's method can be obtained by setting $q = 0$, i.e., by using constant interpolation $p_0 = t_{k+1}$ in the AM process. Besides, with $q = 1$ the AM method is identical with trapezoidal method (2.21) because the resulting quadrature in $[t_k, t_{k+1}]$ is just the well-known trapezoidal rule which is obtained by linear interpolation on points $\{t_k, t_{k+1}\}$. For

$q = 2$ the interpolant p_2 should set up on points $\{t_{k-1}, t_k, t_{k+1}\}$ and integration should apply on interval $[t_k, t_{k+1}]$. The resulting method is the 2-step AM method listed in the second row of Table 5. Other AM methods up to order 4 are given in this table with the corresponding truncation errors in the last column. As we observe, the h powers in local truncation errors are the same as their counterparts in the table of AM methods. However, the constants behind the h are remarkably smaller in the AM methods. The error analysis is given in section 6.4 below.

Table 5: Adams-Moulton methods. Here by y'_k we mean $f(t_k, y_k)$.

q	Order	Method	τ_{k+1}
0	1	$y_{k+1} = y_k + hy'_{k+1}$	$-\frac{1}{2}h^2 y''(\xi_k)$
1	2	$y_{k+1} = y_k + \frac{1}{2}h[3y'_{k+1} - y'_k]$	$-\frac{1}{12}h^3 y'''(\xi_k)$
2	3	$y_{k+1} = y_k + \frac{1}{12}h[5y'_{k+1} + 8y'_k - y'_{k-1}]$	$-\frac{1}{24}h^4 y^{(4)}(\xi_k)$
3	4	$y_{k+1} = y_k + \frac{1}{24}h[9y'_{k+1} + 19y'_k - 5y'_{k-1} + y'_{k-2}]$	$-\frac{19}{720}h^5 y^{(5)}(\xi_k)$

Discussion on using other methods with consistent accuracies for calculating few initial values to bootstrap the AB methods is applicable here for the AM methods as well.

Since AM methods are implicit, an initial guess $y_{k+1}^{(0)}$ and an iteration on y_{k+1} are needed in each time step. A choice for initial guess can be a solution of an AB method of the same order. For example, to implement the AM method of order 3 we may write

$$y_{k+1}^{(0)} = y_k + \tfrac{1}{12}h[23y'_k - 16y'_{k-1} + 5y'_{k-2}],$$
$$y_{k+1}^{(1)} = y_k + \tfrac{1}{12}h[5f(t_{k+1}, y_{k+1}^{(0)}) + 8y'_k - y'_{k-1}].$$

If h is sufficiently small, we can accept $y_{k+1}^{(1)}$ as the solution y_{k+1}. Otherwise, we can proceed with more iterations at the expense of additional evaluations of $f(t, y)$.

6.3 Backward differentiation formulas (BDF)

Another class of efficient numerical methods with excellent stability properties is the backward differentiation formulas (BDF). As the name implies, they are backward (implicit) formulas. For constructing a BDF of order q, we assume that p is a polynomial of degree q that interpolates $y(t)$ at points $\{t_{k+1}, t_k, \ldots, t_{k-q+1}\}$ for $k \geqslant q - 1$:

$$p(t) = \sum_{j=-1}^{q-1} \ell_j(t) y(t_{k-j}) \tag{6.5}$$

where $\ell_j(t), j = -1, \ldots, q-1$ are Lagrange polynomials on points $\{t_{k+1}, t_k, \ldots, t_{k-q+1}\}$. Then we use

$$p'(t_{k+1}) \approx y'(t_{k+1}) = f(t_{k+1}, y(t_{k+1})). \tag{6.6}$$

Since the interpolation points are equidistance with distance h, we can use the change of variable

$$\theta = \frac{t - t_{k+1}}{h}$$

to simplify the notation. This change of variable maps the interpolation points to integer set $\{0, -1, -2, \dots, -(k-q)\}$. In particular, $t = t_{k+1}$ is mapped to $\theta = 0$. If Lagrange functions on this scaled points are denoted by $\tilde{\ell}_j(\theta)$, we can simply show that

$$\ell_j(t) = \tilde{\ell}_j(\theta) \quad \text{and} \quad \ell_j'(t) = \frac{1}{h}\tilde{\ell}_j(\theta).$$

On the other hand, the Lagrange interpolation error is

$$\begin{aligned} e(t) =& \frac{(t-t_{k+1})(t-t_k)\cdots(t-t_{k-q+1})}{(q+1)!} y^{(q+1)}(\zeta_k(t)) \\ =& \frac{h^{q+1}\theta(\theta-1)\cdots(\theta-k+q)}{(q+1)!} y^{(q+1)}(\zeta_k(t)) \end{aligned}$$

for some $t_{k-q+1} \leqslant \zeta_k(t) \leqslant t_{k+1}$. The error of differentiation in (6.6) at $t = t_{k+1}$ (or $\theta = 0$) then is

$$e'(t_{k+1}) = \frac{1}{h}\frac{h^{q+1}(-1)^{k-q}(k-q)!}{(q+1)!} y^{(q+1)}(\xi_k) =: \tilde{\tau}_{k+1}, \tag{6.7}$$

where $\xi_k = \zeta_k(t_{k+1})$. Combining (6.5) and (6.6) and adding the error term give

$$\frac{1}{h}\tilde{\ell}_{-1}'(0)y(t_{k+1}) + \frac{1}{h}\sum_{j=0}^{q-1}\tilde{\ell}_j'(0)y(t_{k-j}) = f(t_{k+1}, y(t_{k+1})) + \tilde{\tau}_{k+1}.$$

By setting

$$\beta = \frac{1}{\tilde{\ell}_{-1}'(0)}, \quad \alpha_j = -\frac{\tilde{\ell}_j'(0)}{\tilde{\ell}_{-1}'(0)}, \quad j = 0, \dots, q-1, \tag{6.8}$$

we obtain

$$y(t_{k+1}) = \sum_{j=0}^{q-1}\alpha_j y(t_{k-j}) + h\beta f(t_{k+1}, y(t_{k+1})) + h\tilde{\tau}_{k+1}.$$

By replacing the exact values $y(t_k)$ by approximate values y_k when dropping the truncation error

$$\tau_{k+1} = h\tilde{\tau}_{k+1} = \frac{(-1)^{k-q}(k-q)!}{(q+1)!} h^{q+1} y^{(q+1)}(\xi_k),$$

the q-step BDF method is obtained as

$$y_{k+1} = \sum_{j=0}^{q-1}\alpha_j y_{k-j} + h\beta f(t_{k+1}, y_{k+1}), \quad k = q-1, q, q+1, \dots. \tag{6.9}$$

The coefficients β and α_j can be simply computed (for example using a symbolic toolbox such as Maple) from (6.8). In Table 6 the q-step BDF methods for $q = 1, 2, \dots, 5$ are given.

The case $q = 1$ is simply the implicit Euler's method (2.14). All discussions concerning the initial values $y_1, \dots, y_{q-1}$ and iteration for nonlinearity that we outlined for the Adams-Moulton methods are applicable here for BDF methods.

As the local truncation error for a q-step BDF method behaves as $\mathcal{O}(h^{q+1})$, it is predictable that the global error of such method behaves as $\mathcal{O}(h^q)$. A general error analysis is given in section 6.4.

Table 6: BDF methods. Here by y'_{k+1} we mean $f(t_{k+1}, y_{k+1})$.

q	Method	τ_{k+1}
1	$y_{k+1} = y_k + hy'_{k+1}$	$-\frac{1}{2}h^2y''(\xi_k)$
2	$y_{k+1} = \frac{4}{3}y_k - \frac{1}{3}y_{k-1} + \frac{2}{3}hy'_{k+1}$	$-\frac{2}{9}h^3y'''(\xi_k)$
3	$y_{k+1} = \frac{18}{11}y_k - \frac{9}{11}y_{k-1} + \frac{2}{11}y_{k-2} + \frac{6}{11}hy'_{k+1}$	$-\frac{3}{22}h^4y^{(4)}(\xi_k)$
4	$y_{k+1} = \frac{48}{25}y_k - \frac{35}{25}y_{k-1} + \frac{16}{25}y_{k-2} - \frac{3}{25}y_{k-3} + \frac{12}{25}hy'_{k+1}$	$-\frac{12}{625}h^5y^{(5)}(\xi_k)$
5	$y_{k+1} = \frac{300}{137}y_k - \frac{300}{137}y_{k-1} + \frac{200}{137}y_{k-2} - \frac{75}{137}y_{k-3} + \frac{12}{137}y_{k-4} + \frac{60}{137}hy'_{k+1}$	$-\frac{10}{137}h^6y^{(6)}(\xi_k)$

Workout 6.1. Consider solving the linear system of equations $y'(t) = Ay(t) + f(t)$ with initial condition $y(0) = y_0$, given constant matrix $A \in \mathbb{R}^{n\times n}$ and know vector function f. Write down the BDF3 scheme for solving this system.

Lab Exercise 6.2. Let's revisit Example 2.5 (MOL solution of the diffusion equation). Impose the initial condition

$$u_0(x) = \sin(\pi x), \quad 0 \leqslant x \leqslant 1.$$

With a spatial step size of $\Delta x = 0.001$, apply the BDF3 method on the resulting system of ordinary differential equations (ODEs) and compute the PDE solution with steplengths $h = 0.1, 0.05, 0.025, 0.0125$, and 0.00625 up to the final time $t = 0.5$. Plot the errors, compute the convergence orders, and compare them with the theoretical order $q = 3$. To initiate BDF3 iterations, in addition to the initial condition $\boldsymbol{y}_0$, we require $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, which should be computed using a one-step method with a truncation error of at least $\mathcal{O}(h^3)$. For instance, the trapezoidal or RK2 methods can be employed for this purpose. The exact solution of this equation with the given initial condition is $u(x,t) = e^{-\pi^2 t}\sin x$, which we can use to compute the errors and orders.
Explain the advantages of BDF3 over explicit Euler, implicit Euler, and RK methods. Recall that when the spatial step size Δx decreases, the size of the ODE system increases and the system becomes stiff.

6.4 Error analysis of multistep methods

In general, a multistep method, including AB, AM, and BDF methods, can be formulated as

$$y_{k+1} = \sum_{j=0}^{q} a_j y_{k-j} + h\sum_{j=-1}^{q} b_j f(t_{k-j}, y_{k-j}), \quad k \geqslant q. \tag{6.10}$$

In both Adams methods (AB and AM methods) $a_0 = 1$ and $a_j = 0$ for $j = 1, \dots, q$. In AB methods $b_{-1} = 0$ and in AM methods $b_q = 0$. In BDF methods $b_{-1} = \beta$, $b_j = 0$ for $j = 0, \dots, q$, and $a_q = 0$.

We present the analysis for the general form (6.10) with only a restrictive condition on coefficients a_j. The analysis will be valid for all three classes of methods mentioned above. It also works for some one-step schemes such as implicit Euler and trapezoidal methods as they are special cases of AM methods. The truncation error for formula (6.10) is defined as

$$\tau_{k+1} = y(t_{k+1}) - \sum_{j=0}^{q} a_j y(t_{k-j}) + h \sum_{j=-1}^{q} b_j y'(t_{k-j}), \quad k \geqslant q. \tag{6.11}$$

For function $\tau(h)$ defined by

$$\tau(h) = \frac{1}{h} \max_{q \leqslant k \leqslant N} |\tau_k|,$$

if we have $\tau(h) \to 0$ as $h \to 0$ then we say the method (6.10) is *consistent*. If

$$\tau(h) = \mathcal{O}(h^m)$$

for some $m \geqslant 1$, we say the order of consistency is m. The following theorem gives necessary and sufficient conditions on coefficients in (6.10) to achieve the consistency.

Theorem 6.3 ([Atkinson-et-al:2009]). Let $m \geqslant 1$ be a given integer. For $\tau(h) \to 0$ for all continuously differentiable function y, that is, for method (6.10) to be consistent, it is necessary and sufficient that

$$\sum_{j=0}^{q} a_j = 1, \tag{6.12}$$

$$-\sum_{j=0}^{q} j a_j + \sum_{j=-1}^{q} b_j = 1. \tag{6.13}$$

Furthermore, to have the consistency order m, i.e., $\tau(h) = \mathcal{O}(h^m)$ for all functions y that are $m+1$ continuously differentiable, it is necessary and sufficient that (6.12) and (6.13) hold and that

$$\sum_{j=0}^{q} (-j)^i a_j + i \sum_{j=-1}^{q} (-j)^{i-1} b_j = 1, \quad i = 2, 3, \dots, m. \tag{6.14}$$

Proof. For the proof just write the degree m Taylor expansion of $y(t)$ around t_k and manipulate the truncation error (6.11). It is left as an exercise. ■

Workout 6.4. Write the proof of Theorem 6.3

The following theorem proves the convergence of the multistep method (6.10). The theorem does not cover all the multistep methods but includes all methods we considered so far such as AB, AM and BDF schemes.

Theorem 6.5 ([Atkinson-et-al:2009]). Consider solving the initial value problem $y'(t) = f(t, y(t))$ for $t \geqslant t_0$ with initial condition $y(t_0) = y_0$ using the multistep method (6.10). Assume that $f(t, y)$ is continuous and satisfies the Lipschitz condition with respect to variable y with Lipschitz constant $L > 0$. Assume that the initial errors satisfy

$$\eta(h) := \max_{0\leqslant k\leqslant q} |y(t_k) - y_k| \to 0, \text{ as } h \to 0.$$

Moreover, assume that y is continuously differentiable and the method is consistent, that is, $\tau(h) \to 0$ as $h \to 0$. Finally, assume that all coefficients a_j are nonnegative:

$$a_j \geqslant 0, \quad j = 0, 1, \ldots, q. \tag{6.15}$$

Then the method (6.10) is convergent and

$$\max_{0\leqslant k\leqslant N} |y(t_k) - y_k| \leqslant c_1\eta(h) + c_2\tau(h) \tag{6.16}$$

for some suitable constants c_1 and c_2. If the solution $y(t)$ is $m+1$ continuously differentiable, the method (6.10) is of consistency order m, and the initial errors satisfy $\eta(h) = \mathcal{O}(h^m)$ then the convergence order is m, i.e., the error is of size $\mathcal{O}(h^m)$.

Proof. Let $e_k = y(t_k) - y_k$. By subtracting (6.10) from (6.11), we obtain

$$e_{k+1} = \sum_{j=0}^{q} a_j e_{k-j} + h \sum_{j=-1}^{q} b_j \Big[f(t_{k-j}, y(t_{k-j})) - f(t_{k-j}, y_{k-j}) \Big] + \tau_{k+1}.$$

Taking absolute value, using the Lipschitz condition, and using assumption (6.15) yield

$$|e_{k+1}| \leqslant \sum_{j=0}^{q} a_j |e_{k-j}| + hL \sum_{j=-1}^{q} |b_j|\, |e_{k-j}| + |\tau_{k+1}|.$$

By introducing the notation

$$E_k = \max_{0\leqslant i\leqslant k} |e_i|, \quad k = 0, 1, \ldots, N$$

we can write

$$|e_{k+1}| \leqslant E_k \sum_{j=0}^{q} a_j + hLE_{k+1} \sum_{j=-1}^{q} |b_j| + h\tau(h).$$

From (6.12) we know that the sum of a_js is 1, thus we have

$$|e_{k+1}| \leqslant E_k + hcE_{k+1} + h\tau(h), \quad c = L \sum_{j=-1}^{q} |b_j|.$$

Since the right hand side is trivially a bound for E_k and $E_{k+1} = \max\{|e_{k+1}|, E_k\}$, we simply have

$$E_{k+1} \leqslant E_k + hcE_{k+1} + h\tau(h).$$

For $hc \leqslant \frac{1}{2}$, which is true as $h \to 0$, we obtain

$$E_{k+1} \leqslant \frac{E_k}{1 - hc} + \frac{h}{1 - hc}\tau(h) \leqslant (1 + 2hc)E_k + 2h\tau(h).$$

If we proceed as in the analysis of the Euler's method in section 2.1.1 we finally have

$$E_N \leqslant e^{2c(b-t_0)}\eta(h) + \left[\frac{e^{2c(b-t_0)} - 1}{c} \right] \tau(h),$$

which complete the proof. ■

To obtain the rate of convergence of $\mathcal{O}(h^m)$ it is necessary that $\tau_{k+1} = \mathcal{O}(h^{m+1})$ in each step k (i.e., $\tau(h) = \mathcal{O}(h^m)$ which needs relation (6.14) to hold), and the initial values $y_0, y_1, \dots, y_q$ need to be computed with an accuracy of only $\mathcal{O}(h^m)$, i.e., $\eta(h) = \mathcal{O}(h^m)$.

6.5 Stability regions of multistep methods

Again consider the general multistep method (6.10). If we apply it on test equation $y'(t) = \lambda y(t)$ we obtain

$$y_{k+1} = \sum_{j=0}^{q} a_j y_{k-j} + h\lambda \sum_{j=-1}^{q} b_j y_{k-j}.$$

Letting $z = \lambda h$ and rearranging the above equation give

$$(1 - zb_{-1})y_{k+1} - \sum_{j=0}^{q}(a_j + zb_j)y_{k-j} = 0. \tag{6.17}$$

This is a homogenous linear *difference equation* of order $q+1$. For absolute stability, we need to find the general solution y_k and impose some condition on z in order to have a bounded $|y_k|$ as $k \to \infty$. The general theory for solving linear difference equation is given in the Appendix A. The general solution can be found by looking for solutions of the special form

$$y_k = r^k, \quad k \geqslant 0.$$

Substituting this special solutions in to (6.18) and cancelling the factor r^{k-q} yield

$$(1 - zb_{-1})r^{q+1} - \sum_{j=0}^{q}(a_j + zb_j)r^{q-j} = 0. \tag{6.18}$$

This equation is called the *characteristic equation.* For example, the characteristic equations for 3-step AB, AM, and BDF methods are

$$\begin{aligned}
&r^3 - (1 + \tfrac{23}{12}z)r^2 + \tfrac{16}{12}zr - \tfrac{5}{12}z = 0,\\
&(1 - \tfrac{9}{24}z)r^3 - (1 + \tfrac{19}{24}z)r + \tfrac{5}{24}zr - \tfrac{1}{24}z = 0,\\
&(1 - \tfrac{6}{11}z)r^3 - \tfrac{18}{11}r^2 + \tfrac{9}{11}r - \tfrac{2}{11} = 0,
\end{aligned}$$

respectively.

Assume that the characteristic polynomial has roots $r_0, r_1, \dots, r_q$. As the roots depend on z, we denote them by $r_0(z), r_1(z), \dots, r_q(z)$. The boundary of the stability region is where all roots have magnitude 1 or less, and at least one root has magnitude 1. Roots with magnitude 1 can be represented as $r = e^{i\theta}$ for $0 \leqslant \theta < 2\pi$, where i is the imaginary number. To obtain the boundary of the stability region we can find all z where (6.18) holds true with $r = e^{i\theta}$. One can separate the terms containing z and write (6.18) in the form

$$\rho(r) - z\sigma(r) = 0$$

where

$$\rho(r) = r^{q+1} - \sum_{j=0}^{q} a_j r^{q-j}, \quad \sigma(r) = b_{-1}r^{q+1} + \sum_{j=0}^{q} b_j r^{q-j}.$$

Now, for all $\theta \in [0, 2\pi)$ we compute the complex values

$$z = \frac{\rho(e^{i\theta})}{\sigma(e^{i\theta})} =: u + iv$$

and plot v against u. The plot includes the boundary of the stability region. With a little care we can identify the true stability region.

Stability regions of the Adams-Bashforth and Adams-Moulton methods are plotted in Figures 21 and 22. It can be seen that for AB and AM methods of equal order, the AM method has a larger absolute stability region. Unless the AM methods of order 1 and 2 which are the implicit Euler's and the trapezoidal methods, stability regions of other AB and AM methods do not contain the left-half plane and even do not encompass the whole negative real line. Consequently, these methods are not adequate for solving stiff differential equations.

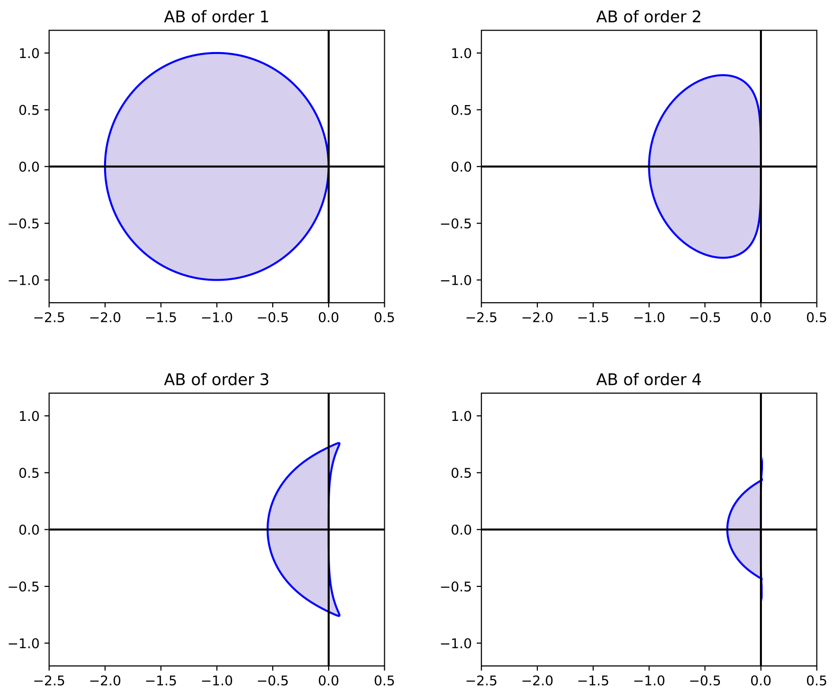


Figure 21: Stability regions of Adams-Bashforth methods of orders 1 to 4.

The stability regions of BDF schemes are plotted in Figure 23. Notably, BDF1 (implicit Euler's method) and BDF2 are A-stable. As the order of the method increases, the region of absolute stability diminishes, although it still encompasses the entire negative real line and maintains $A(\alpha)$-stability. The angle α decreases with increasing order from 1 to 6. Nonetheless, BDF1 to BDF6 are suitable schemes for solving stiff differential equations. However, schemes of orders $q \geqslant 7$ lose this property and are not adequate for solving stiff problems.

6.6 One-step versus multistep methods

Finally, we compare one-step and multistep methods for numerical solution of ODEs. Some advantages of one-step methods over multistep methods are listed below.

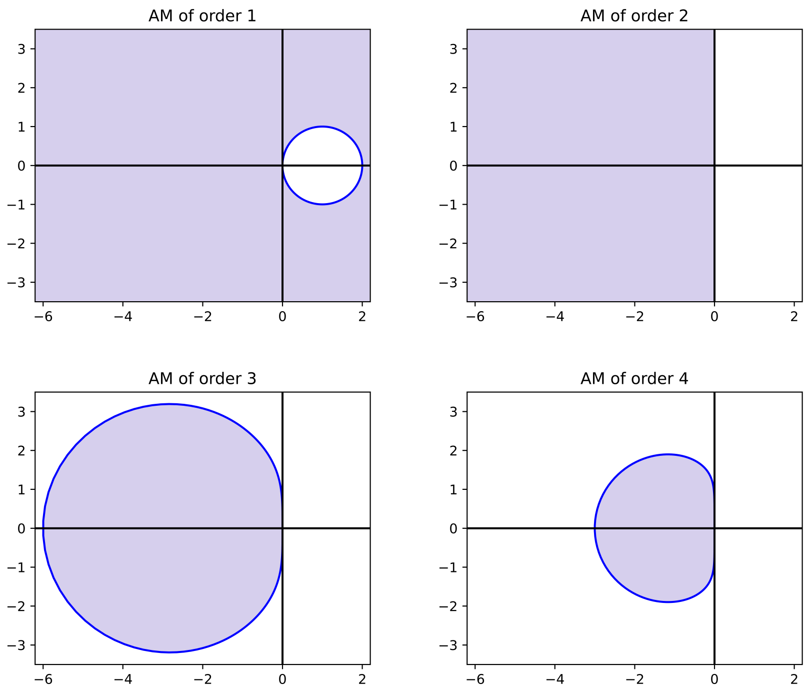


Figure 22: Stability regions of Adams-Moulton methods of orders 1 to 4. Note the different scale on the axes comapred to Figure 21.

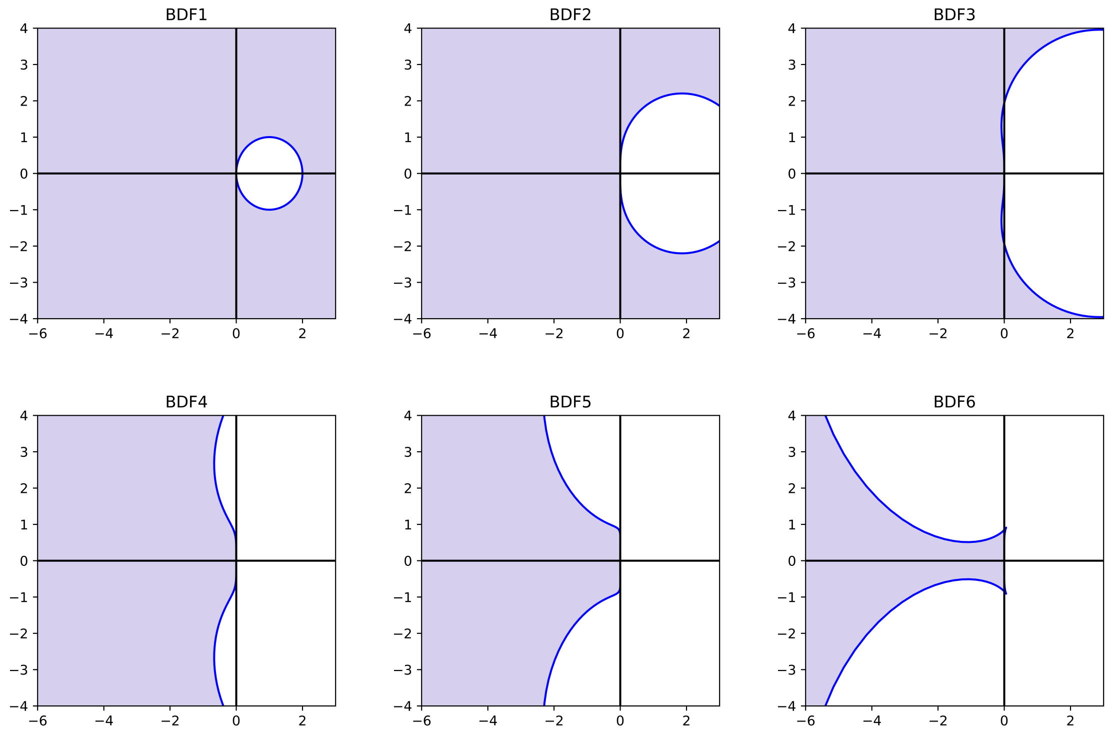


Figure 23: Stability regions of BDF methods of orders 1 to 6. BDF1 and BDF2 are A-stable and BDF3-BDF6 are $A(\alpha)$-stable.

- One-step methods are self-starting: the available initial guess y_0 is enough to start and compute next values $y_1, y_2, \ldots$.

- Adaptivity is easier with one-step methods. The step length h can be changed at any point in one-step method, based on an error estimate. In a multistep methods, however, more care is required since the previous values are assumed to be equally spaced in the standard form of these methods given in Table 4 and 5.
- If the solution $y(t)$ is not smooth at some isolated point t^* then with a one-step method it is often possible to get full accuracy simply by ensuring that t^* is a grid point. This is impossible with multistep methods.

On the other hand, one-step methods have some disadvantages. The disadvantage of Taylor series methods is that they require differentiating the given equation and are cumbersome and often expensive to implement. RK methods only use evaluations of the function f, but a higher order RK method requires evaluating f several times each time step. While, in multistep methods only one new f evaluation is required in each time step.

7 MATLAB's ODE suite

A suite of ODE solvers was introduced with version 5 of MATLAB and continued to current versions. The suite now contains seven solvers. Here we introduce some of them.

- `ode23`

  This is a low-order adaptive embedded solver for *nonstiff* ODEs. The '23' in the function name indicates that two simultaneous single-step formulas, one of second order and one of third order, are involved.

- `ode45`

  This is a high order embedded Runge-Kutta-Fehlberg solver which uses a fourth-order and fifth-order formulas. For differential equations with smooth solutions, `ode45` is often more accurate than `ode23` but still works for *nonstiff* ODEs. The MATLAB documentation recommends `ode45` as the first choice and Simulink blocks set `ode45` as the default solver.

- `ode23t`

  This solver is an implementation of the trapezoidal rule using a free interpolant. Use this solver if the problem is only *moderately stiff* and you need a solution without numerical damping. This solver can solve differential algebraic equations (DAEs) as well.

- `ode15s`

  This solver is suitable for solving *stiff* ODEs and DAEs. It is a variable order solver based on the numerical differentiation formulas. Optionally, it uses the backward differentiation formulas (BDFs, also known as Gear's method) which are a class of multistep solvers. (we did not learn multistep methods in this course). Try `ode15s` when `ode45` fails, or is very inefficient.

- See also `ode113`, `ode78`, `ode89`, etc. in the MATLAB's help document.

You can explore other ODE and DAE solvers by referring to MATLAB's help documentation or other sources available. The syntax for calling these solvers is one of the

```
[t,y] = solver(odefun,tspan,y0)
[t,y] = solver(odefun,tspan,y0,options)
```

where `solver` is one of `ode45`, `ode23` and else. `odefun` is a function that evaluates $f(t, y)$, the right-hand side of the differential equations. `y0` is the initial condition. `tspan` is a two-vector specifying the interval of integration, $[t_0, t_{final}]$. To obtain solutions at specific times (all increasing or all decreasing), use `tspan` $= [t_0, t_1, \dots, t_{final}]$. For example the command

```
[t,y] = ode45(odefun,[0:0.01:0.5],[0 1]);
```

returns the function values at the specified vector `[0:0.01:0.5]`. But if the values at specified points are not required you can simply set `tspan = [0 0.5]`.

Optional integration arguments are created using the `odeset` function. For example we can customize the error tolerances using

```
options = odeset('RelTol',1e-6,'AbsTol',1e-10);
[t,y] = ode45(@myfunc,[0 0.5],[0 1],options);
```

This guarantees that the error at each step is less than `RelTol` times the value at that step, and less than `AbsTol`. More precisely

$$|e_k| \leqslant \max\{\texttt{RelTol} \times |y_k|, \texttt{AbsTol}\}.$$

Decreasing error tolerance can considerably slow the solver.

Example 7.1. To solve the scaler value equation

$$y'(t) = t^3/y(t), \quad 0 \leqslant t \leqslant 10, \quad y(0) = 1$$

using `ode23` we can write (without additional options):

```
[t,y] = ode23(@(t,y) t^3/y,[0 10],1);
plot(t,y,'-ob')
```

The second example is a known nonstiff system of equations describing the motion of a rigid body without external forces:

$$\begin{aligned} y_1'(t) &= y_2(t)y_3(t) \\ y_2'(t) &= -y_1(t)y_3(t) \\ y_3'(t) &= -0.51y_1(t)y_2(t) \end{aligned}$$

with initial conditions $y_1(0) = 0$, $y_2(0) = 1$ and $y_3(0) = 1$. The time interval is $[0, 12]$. With optional relative and absolute errors, we can compute the solutions by writing the following script.

```
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4 1e-5]);
[t,y] = ode45(@rigid,[0 12],[0 1 1],options);
plot(t,y(:,1),'-',t,y(:,2),'-.',t,y(:,3),'.');
```

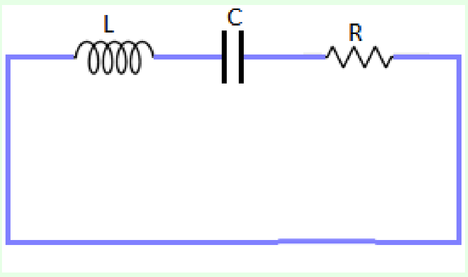
The `rigid` function can be written in a separated script as a new function:

```
function yprime = rigid(t,y)
yprime = zeros(3,1);
yprime(1) = y(2)*y(3);
yprime(2) = -y(1)*y(3);
yprime(3) = -0.51*y(1)*y(2);
end
```

All solvers solve systems of equations in the form $\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$ or problems that involve a mass matrix, $M(t, \boldsymbol{y})\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y})$. `ode15s` and `ode23t` can solve problems with a mass matrix that is singular, i.e., DAEs. The mass matrix can be imported via `odeset`. See doc `odeset` in the MATLAB's help for a list of options you can customize.

Lab Exercise 7.1. Repeat solving examples in Lab Exercise 2.3 using `ode23` and `ode45`. Try to test different options. In each case plot the exact and numerical solutions in the same figure.

Lab Exercise 7.2. The Van der Pol oscillator is a self-maintained electrical circuit comprised of an inductor (L), an initially charged capacitor with capacitance (C), and a nonlinear resistor (R), all connected in series, as shown in the figure below.



By using the operational amplifier, the characteristic intensity-tension of the nonlinear resistance (R) is given as

$$U_R = -R_0 i_0 \left[\frac{i}{i_0} - \frac{1}{3} \left(\frac{i}{i_0} \right)^3 \right] \tag{7.1}$$

where i is the current and i_0 and R_0 are the current and the resistance of the normalization respectively. By applying the Kirchhoff's law to the above figure we have

$$U_L + U_R + U_C = 0 \tag{7.2}$$

where U_L and U_C are the tension to the limits of the inductor and capacitor, respectively, and are defined as

$$U_L = L \frac{\mathrm{d}i}{\mathrm{d}t}, \quad U_C = \frac{1}{C} \int i \, \mathrm{d}t. \tag{7.3}$$

1. Substitute (7.1) and (7.3) into (7.2) and obtain an integral-differential equation. Then differentiate it and obtain the following second order ODE:

   $$L \frac{\mathrm{d}^2 i}{\mathrm{d}\tau^2} - R_0 \left[1 - \frac{i^2}{i_0^2} \right] \frac{\mathrm{d}i}{\mathrm{d}\tau} + \frac{i}{C} = 0. \tag{7.4}$$

   Then use the change of variables $u = i/i_0$ and $t = \omega\tau$ where $\omega = 1/\sqrt{LC}$ (electric pulsation), to convert (7.4) to the following equation:

   $$\frac{\mathrm{d}^2 u}{\mathrm{d}t^2} - R_0 \sqrt{\frac{C}{L}} \left(1 - u^2 \right) \frac{\mathrm{d}u}{\mathrm{d}t} + u = 0.$$

2. By setting $\mu = R_0 \sqrt{\frac{C}{L}}$ and adding initial conditions obtain the well-known *Van der Pol* equation

   $$\begin{aligned} &u'' - \mu u'(1 - u^2) + u = 0, \quad 0 \leqslant t \leqslant b \\ &u(0) = u_0, \quad u'(0) = u_0'. \end{aligned} \tag{7.5}$$

   and convert it to a system of first-order ODEs.

3. Apply RK2 and RK4 methods (your own codes) for solving the system obtained from equation (7.5) with different values $\mu = 1, 10, 100, 1000$, and with initial conditions $u_0 = 2$ and $u_0' = 0$. For $\mu = 1$ let $b = 20$, for $\mu = 10$ let $b = 100$, for $\mu = 100$ let $b = 500$ and for $\mu = 1000$ let $b = 5000$. For large values of μ use a supper small stepsize h to get accurate results. In each case plot the numerical solution u and u' in terms of t in interval $[0, b]$ and report your observations. Also report the executing times (sec.) in a table.

4. Now use ODE solvers `ode45`, `ode23t` and `ode15s` to solve this ODE again with different values of μ and b given in item (1). In each case produce the plot of u and u' in terms of t, and compute the CPU time required (sec.) for executing the codes. Compare with the results of item (2).

8 Appendix

A: Difference equations

Consider the following *homogeneous difference equation* of order n,

$$c_k y_k + c_{k-1} y_{k-1} + \cdots + c_{k-p} y_{k-p} = 0, \quad k \geqslant p \tag{8.1}$$

with given initial values $y_0, y_1, \dots, y_{p-1}$. The general solution of this equation is obtained by looking for solutions of the special form

$$y_k = r^k, \quad k \geqslant 0.$$

If we can obtain p linearly independent solutions, then an arbitrary linear combinations of this solutions give the general solution of (8.1). Substituting $y_k = r^k$ into (8.1) and cancelling r^{k-p}, we obtain

$$c_p r^p + c_{p-1} r^{p-1} + \cdots + c_1 r + c_0 = 0 \tag{8.2}$$

which is called *characteristic equation*, and the left side is *characteristic polynomial.* If (8.2) possesses p distinct solutions (roots) $r_1, r_2, \dots, r_p$ then the general solution of (8.1) is

$$y_k = \sum_{j=1}^{p} \beta_j r_j^k, \quad k \geqslant 0. \tag{8.3}$$

The coefficients $\beta_0, \beta_1, \dots, \beta_{p-1}$ are obtained by imposing the known initial values $y_0, y_1, \dots, y_{p-1}$ into the general solution (8.3). It is clear that if $|r_j| \leqslant 1$ then the solution y_k does not grow as $k \to \infty$. However, if r_j is a root of multiplicity ν, i.e.,

$$r_j = r_{j+1} = \cdots = r_{j+\nu-1},$$

then the ν linearly independent solutions corresponding to these roots are

$$r_j^k,\ k r_j^k, \dots, k^{\nu-1} r_j^k$$

and in formula (8.3) the part $\beta_j r_j^k + \beta_{j+1} r_{j+1}^k + \cdots + \beta_{j+\nu-1} r_{j+\nu-1}^k$ should be replaced by

$$[\beta_j + k\beta_{j+1} + \cdots + k^{\nu-1}\beta_{j+\nu-1}] r_j^k.$$

In this case the solution y_k remains stable as $k \to \infty$ provided that $|r_j| \leqslant 1$ for simple roots and $|r_j| < 1$ for roots with multiplicity.

Example 8.1. The general solution of difference equation

$$y_k + 5y_{k-1} + 6r_{k-2} = 0$$

is obtained in terms of roots $r_1 = 2$ and $r_2 = 3$ of characteristic polynomial $r^2 + 3r + 2$,

$$y_k = \beta_1 2^k + \beta_2 3^k.$$

If $y_0 = 0$ and $y_1 = 2$ are given then by solving $y_0 = \beta_1 + \beta_2$ and $y_1 = 2\beta_1 + 3\beta_2$ we simply obtain $\beta_1 = -2$ and $\beta_2 = 2$. The solution then is

$$y_k = -2 \times 2^k + 2 \times 3^k, \quad k = 0, 1, 2, 3, \dots.$$

As second example consider the following difference equation

$$y_k - 5y_{k-1} + 6y_{k-2} + 4y_{k-3} - 8y_{k-4} = 0.$$

The characteristic equation is $r^4 - 5r^3 + 6r^2 + 4r - 8 = 0$ with roots $r_1 = -1$ and $r_2 = r_3 = r_4 = 2$. The general solution then is

$$y_k = \beta_1(-1)^k + [\beta_2 + k\beta_3 + k^2\beta_4]2^k.$$

With given initial values $y_0 = -1$, $y_1 = y_2 = -7$ and $y_3 = 7$ we must solve

$$\begin{aligned}
-1 &= \beta_1 + \beta_2 \\
-7 &= -\beta_1 + 2[\beta_2 + \beta_3 + \beta_4] \\
7 &= \beta_1 + 4[\beta_2 + 2\beta_3 + 4\beta_4] \\
7 &= -\beta_1 + 8[\beta_2 + 3\beta_3 + 9\beta_4]
\end{aligned}$$

to obtain coefficients β_j. Solving this system gives $\beta_1 = 1, \beta_2 = -2, \beta_3 = -2$ and $\beta_4 = 1$.

References

[1] K. E. Atkinson, W. Han, D. E. Stewart, *Numerical Solution of Ordinary Differential Equations*, Wiley, 2009.

[2] M. T. Heath, *Scientific Computing, an Introductory Survey*, revised 2nd edition, SIAM, Philadelphia, PA, 2018.

[3] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems*, Wiley, 1991.

[4] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations*, SIAM, 2007.