Numerical solution of ordinary differential equations (ODE)

1 Background

A system of m first order ordinary differential equations is given by

$$\begin{aligned}
x_1' &= f_1(t, x_1, \ldots, x_m), \\
x_2' &= f_2(t, x_1, \ldots, x_m), \\
&\vdots \\
x_m' &= f_m(t, x_1, \ldots, x_m),
\end{aligned}$$

or in vectorform

$$\mathbf{x}' = \mathbf{f}(t, \mathbf{x}),$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}, \qquad \mathbf{f}(t, \mathbf{x}), = \begin{bmatrix} f_1(t, x_1, \ldots, x_m) \\ f_2(t, x_1, \ldots, x_m) \\ \vdots \\ f_m(t, x_1, \ldots, x_m) \end{bmatrix}.$$

This is called an *initial value problem* (IVP) if the solution is given at a point t_0,

$$\begin{aligned}
x_1(t_0) &= x_{1,0}, \\
x_2(t_0) &= x_{2,0}, \\
&\vdots \\
x_m(t_0) &= x_{m,0}.
\end{aligned}$$

Example: *An example of the Lotka-Volterra equations (Predator/prey-model)*

$$\begin{aligned}
x_1' &= x_1 - x_1 x_2, \\
x_2' &= x_1 x_2 - 2x_2.
\end{aligned}$$

1.1 Autonomous system

An ODE is called *autonomous* if f isn't a function of t, but only of $\mathbf{x}$. The Lotka-Volterra equations is an example of an autonomous system. A non-autonomous system can be made autonomous by adding an additional equation. By letting $x_0 = t$ this leads to

$$x_0' = 1, \qquad x_0(t_0) = t_0.$$

We end up with a system of $m + 1$ equations where t is substituted by x_0.

1.2 Higher order equations

A higher order initial value problem is given by

$$\begin{aligned} x^{(m)} &= f(t, x, x', x'', \dots, x^{(m-1)}), \\ x(t_0) &= x_0, \\ x'(t_0) &= x'_0, \\ &\vdots \\ x^{(m-1)}(t_0) &= x_0^{(m-1)}, \end{aligned}$$

where $x^{(m)} = \mathrm{d}^m x/\mathrm{d}t^m$. This can be written as a system of 1.order equations by introducing

$$\begin{aligned} x_1 &= x, \\ x_2 &= x', \\ &\vdots \\ x_m &= x^{(m-1)}. \end{aligned}$$

This leads to the system

$$\begin{aligned} \mathbf{x}' &= \mathbf{f}(t, \mathbf{x}), \\ \mathbf{x}(t_0) &= \mathbf{x}_0, \end{aligned}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{m-1} \\ x_m \end{bmatrix}, \qquad \mathbf{f}(t, \mathbf{x}) = \begin{bmatrix} x_2 \\ x_3 \\ \vdots \\ x_m \\ f(t, x_1, \dots, x_m) \end{bmatrix}, \qquad \mathbf{x}_0 = \begin{bmatrix} x_0 \\ x'_0 \\ \vdots \\ x_0^{(m-2)} \\ x_0^{(m-1)} \end{bmatrix}.$$

We will conclude this section with some existence and uniqueness results.

Definition: *A function $\mathbf{f} : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ satisfies the Lipschitz condition with respect to $\mathbf{x}$ in a domain $(a, b) \times D$, where $D \subset \mathbb{R}^m$, if there exists a constant L such that*

$$\|\mathbf{f}(t, \mathbf{x}) - \mathbf{f}(t, \tilde{\mathbf{x}})\| \le L \|\mathbf{x} - \tilde{\mathbf{x}}\|, \qquad \text{for all} \quad t \in (a, b), \quad \mathbf{x}, \tilde{\mathbf{x}} \in D.$$

L is called the Lipschitz constant.

It can be shown that the function $\mathbf{f}$ satisfies the Lipschitz condition if $\partial f_i / \partial x_j$, $i, j = 1, \dots, m$, are continuous and bounded in the domain.

Theorem: *We have the initial value problem*

$$\mathbf{x}' = \mathbf{f}(t, \mathbf{x}), \quad \mathbf{x}(t_0) = \mathbf{x}_0. \tag{1}$$

If

1. $\mathbf{f}(t, \mathbf{x})$ is continuous in $(a, b) \times D$,

2. $\mathbf{f}(t, \mathbf{x})$ satisfies the Lipschitz condition with respect to $\mathbf{x}$ in $(a, b) \times D$,

with given initial values $t_0 \in (a, b)$ and $\mathbf{x}_0 \in D$, then (1) has one and only one solution in $(a, b) \times D$.

2 Numerical solution of ODEs

In this section we will consider some simple methods for solving initial value problems. We still consider the problem (1) and we will assume that we have solutions $\mathbf{x}_l \simeq \mathbf{x}(t_l), \quad l = 0, 1, \dots, n$, and we want to find an approximation $\mathbf{x}_{n+1} \simeq \mathbf{x}(t_{n+1})$, where $t_{n+1} = t_n + h$, and h is the stepsize. We will consider two classes for solving such problems:

1. *Onestep methods*: Only $\mathbf{x}_n$ is used to find the approximation $\mathbf{x}_{n+1}$. Such methods typically require more than one function evaluation pr. step. They can be written on the form
$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\Phi(t_n, \mathbf{x}_n; h)$$
2. *Linear multistep methods*: $\mathbf{x}_{n+1}$ is approximated by using $\mathbf{x}_{n-k+1}$, with different values $k \geq 0$.

2.1 Some examples of onestep methods

Assume that we know $t_n, \mathbf{x}_n$. The exact solution $\mathbf{x}(t_{n+1})$ of (1) is given by

$$\mathbf{x}(t_n + h) = \mathbf{x}_n + \int_{t_n}^{t_{n+1}} \mathbf{x}'(\tau)\mathrm{d}\tau = \mathbf{x}_n + \int_{t_n}^{t_{n+1}} \mathbf{f}(\tau, \mathbf{x}(\tau))\mathrm{d}\tau.$$

Hence, we need an approximation to the last integral. The simplest is to use $\mathbf{f}(\tau, \mathbf{x}(\tau)) \simeq \mathbf{f}(t_n, \mathbf{x}_n)$ which leads to Euler's method:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{f}(t_n, \mathbf{x}_n).$$

Another alternative is to approximate the integral by using the trapezoidal rule:

$$\int_{t_n}^{t_{n+1}} \mathbf{f}(\tau, \mathbf{x}(\tau))\mathrm{d}\tau \simeq \frac{h}{2}(\mathbf{f}(t_n, \mathbf{x}_n) + \mathbf{f}(t_{n+1}, \mathbf{x}_{n+1})).$$

This leads to the numerical scheme:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}(\mathbf{f}(t_n, \mathbf{x}_n) + \mathbf{f}(t_{n+1}, \mathbf{x}_{n+1})). \tag{2}$$

In order to find $\mathbf{x}_{n+1}$ we need to solve a (generally) non-linear system. Such methods are called *implicit methods*, and are often more expensive pr. time-step than corresponding *explicit methods*. The main reason for choosing an implicit method is related to numerical stability, which is particularly relevant for solving so-called *stiff problems*.

A possible modification of (2) is to approximate $\mathbf{x}_{n+1}$ on the right-hand side by using Euler's method

$$\begin{aligned}\tilde{\mathbf{x}}_{n+1} &= \mathbf{x}_n + h\mathbf{f}(t_n, \mathbf{x}_n),\\ \mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{2}(\mathbf{f}(t_n, \mathbf{x}_n) + \mathbf{f}(t_{n+1}, \tilde{\mathbf{x}}_{n+1})).\end{aligned}$$

This method is explicit and is called *Heun's method*.

3 Runge-Kutta methods

Euler's method and Heun's method are both examples of *explicit Runge-Kutta methods* (ERK). Such schemes are given by

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{x}_n), \\
\mathbf{k}_2 &= \mathbf{f}(t_n + c_2 h, \mathbf{x}_n + h a_{21} \mathbf{k}_1), \\
\mathbf{k}_3 &= \mathbf{f}((t_n + c_c h, \mathbf{x}_n + h(a_{31}\mathbf{k}_1 + a_{32}\mathbf{k}_2)), \\
&\vdots \\
\mathbf{k}_s &= \mathbf{f}(t_n + c_s h, \mathbf{x}_n + h \sum_{j=1}^{s-1} a_{sj} \mathbf{k}_j), \\
\mathbf{x}_{n+1} &= \mathbf{x}_n + h \sum_{i=1}^{s} b_i \mathbf{k}_i,
\end{aligned}$$

where c_i, a_{ij} and b_i are coefficients which defines the method. We always require $c_i = \sum_{j=1}^{s} a_{ij}$. Here, s is called the number of *stages*.

Heun's method is a two-stage RK-method, given by

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{x}_n), \\
\mathbf{k}_2 &= \mathbf{f}(t_n + h, \mathbf{x}_n + h\mathbf{k}_1), \\
\mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2).
\end{aligned}$$

Implict methods, such as the trapeziodal method

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \frac{h}{2}(\mathbf{f}(t_n, \mathbf{x}_n) + \mathbf{f}(t_{n+1}, \mathbf{x}_{n+1})).$$

may also be written in a similar form

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{x}_n), \\
\mathbf{k}_2 &= \mathbf{f}(t_n + h, \mathbf{x}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2)), \\
\mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2).
\end{aligned}$$

Here, we need to solve a (non-linear) system in order to find $\mathbf{k}_2$.

Definition: *An s-stage Runge-Kutta method is given by*

$$\begin{aligned}
\mathbf{k}_i &= \mathbf{f}(t_n + c_i h, \mathbf{x}_n + h \sum_{j=1}^{s} a_{ij} \mathbf{k}_j), \qquad i = 1, 2, \ldots, s, \\
\mathbf{x}_{n+1} &= \mathbf{x}_n + h \sum_{i=1}^{s} b_i \mathbf{k}_i.
\end{aligned}$$

This method is defined by the coefficients, which are often presented in a Butcher-tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & & & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array}$$

where $c_i = \sum_{i=1}^{s} a_{ij}, \quad i = 1, \dots, s$. *This method is explicit if* $a_{ij} = 0$ *when* $j \geq i$

Example: *Butcher-tableau for the three methods we so far have considered*

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

Euler *Heun* *Trapes*

When the method is explicit the zeros above and on the diagonal are usually not included. We conclude this section by presenting the most known RK-method: the classical 4th order explicit Runge-Kutta method (Kutta - 1901):

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_m n, \mathbf{x}_n), \\ \mathbf{k}_2 &= \mathbf{f}(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{h}{2}\mathbf{k}_1), \\ \mathbf{k}_3 &= \mathbf{f}(t_n + \frac{h}{2}, \mathbf{x}_n + \frac{h}{2}\mathbf{k}_2), \\ \mathbf{k}_4 &= \mathbf{f}(t_n + h, \mathbf{x}_n + h\mathbf{k}_3), \\ \mathbf{x}_{n+1} &= \mathbf{x}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), \end{aligned} \qquad \begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

3.1 Order conditions for Runge-Kutta methods

We have the following theorem:

Theorem: *Let the problem*

$$\mathbf{x}' = \mathbf{f}(t, \mathbf{x}), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad t_0 \leq t \leq t_{\text{end}}$$

be solved by a onestep method

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\Phi(t_n, \mathbf{x}_n; h)$$

with stepsize $h = (t_{end} - t_0)/N$, *where* N *is the number of steps. If*

1. $\Phi(t_n, \mathbf{x}_n; h)$ *is Lipschitz in* $\mathbf{x}$ *and*
2. *the local truncation error is* $\mathbf{d}_{n+1} = \mathcal{O}(h^{p+1})$,

the method is of order p, which means that the global error satisfies

$$\mathbf{e} = \mathbf{x}(t_{\text{end}}) - \mathbf{x}_N = \mathcal{O}(h^p).$$

A RK method is a onestep method with increment function $\Phi(t_n, \mathbf{x}_n; h) = \sum_{i=1}^{s} b_i \mathbf{k}_i$. It is possible to show that Φ is Lipschitz in $\mathbf{x}$ if $\mathbf{f}$ is Lipschitz and $h \leq h_{\max}$, where $h_{\max}$ is some predefined maximum stepsize. What remains is the order of the local truncation error. To find it, we take the Taylor-expansions of the exact and the numerical solutions and compare. The local truncation error is $\mathcal{O}(h^{p+1})$ if the two series matches for all terms corresponding to $h^q, \quad q \leq p$. In principle, this is trivial. In practise, it becomes extremely tedious. The order conditions up to order 4 are given by

Order	Condition
1	$\sum b_i = 1$
2	$\sum b_i c_i = 1/2$
3	$\sum b_i c_i^2 = 1/3$
	$\sum b_i a_{ij} c_j = 1/6$
4	$\sum b_i c_i^3 = 1/4$
	$\sum b_i c_i a_{ij} c_j = 1/8$
	$\sum b_i a_{ij} c_j^2 = 1/12$
	$\sum b_i a_{ij} a_{jk} c_k = 1/24$

4 Linear multistep methods

An alternative strategy for solving problems on the form (1) is by using multistep methods.

A linear multistep method with s stepsis on the form

$$\sum_{j=0}^{s} \alpha_j \mathbf{x}_{k+j} = h \sum_{j=0}^{s} \beta_j \mathbf{f}(t_{k+j}, \mathbf{x}_{k+j}), \tag{3}$$

where $\alpha_s \neq 0$. If $\beta_s \neq 0$ the right-hand side includes $\mathbf{x}_{k+s}$ and the method is *implicit*; if $\beta_s = 0$ the method is *explicit*.

An advantage with such methods is that functionevaluations from previous time-steps may be reused, which wasn't the case for the onestep-methods in the previous section. A disadvantage is that it is more complicated to change the step-size at different time-steps compared to onestep methods.

4.1 Adams-Bashforth methods

Adams-Bashforth methods is a class of explicit multistep methods (3) with $\alpha_0 = \alpha_1 = \ldots = \alpha_{s-2} = 0$, $\alpha_{s-1} = -1$, $\alpha_s = 1$ and $\beta_s = 0$. Previously we found

$$\mathbf{x}(t_n + h) = \mathbf{x}_n + \int_{t_n}^{t_{n+1}} \mathbf{f}(\tau, \mathbf{x}(\tau)) \mathrm{d}\tau. \tag{4}$$

By approximating the integrand in the last integral as a constant over (t_n, t_{n+1}) we find (see Figure 1):

AB1:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\mathbf{f}(t_n, \mathbf{x}_n).$$

This corresponds to Euler's method. However, if we use $\mathbf{f}(t_{n-1}, \mathbf{x}_{n-1})$ and $\mathbf{f}(t_n, \mathbf{x}_n)$ to approximate $\mathbf{f}(\tau, \mathbf{x}(\tau))$ as a linear function in the interval (t_n, t_{n+1}) we find (see Figure 1):

AB2:

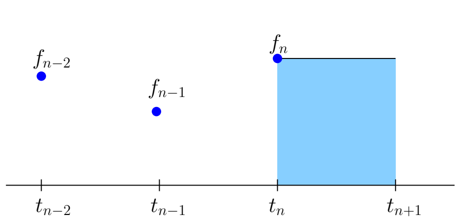
$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\left(\frac{3}{2}\mathbf{f}(t_n, \mathbf{x}_n) - \frac{1}{2}\mathbf{f}(t_{n-1}, \mathbf{x}_{n-1})\right).$$

Similarly, by using $\mathbf{f}(t_{n-2}, \mathbf{x}_{n-2})$, $\mathbf{f}(t_{n-1}, \mathbf{x}_{n-1})$ and $\mathbf{f}(t_n, \mathbf{x}_n)$ to approximate $\mathbf{f}(\tau, \mathbf{x}(\tau))$ as a quadratic function in the interval (t_n, t_{n+1}) we find (see Figure 1):

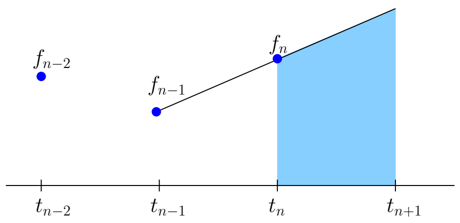
AB3:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\left(\frac{23}{12}\mathbf{f}(t_n, \mathbf{x}_n) - \frac{4}{3}\mathbf{f}(t_{n-1}, \mathbf{x}_{n-1}) + \frac{5}{12}\mathbf{f}(t_{n-2}, \mathbf{x}_{n-2})\right).$$

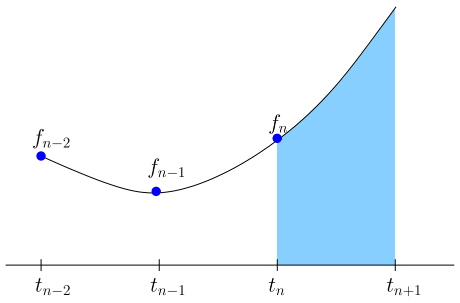
It can be shown that the global error for an Adams-Bashforth method with s steps is $\mathcal{O}(h^s)$.



(a) AB1



(b) AB2



(c) AB3

Figure 1: Approximation of $\mathbf{f}(\tau, \mathbf{x}(\tau))$ for Adams-Bashforth methods.

4.2 Adams-Moulton methods

Adams-Moulton methods is a class of implicit multistep methods. These can also be derived by approximating the integrand in (4), but now we also use $\mathbf{f}(t_{n+1}, \mathbf{x}_{n+1})$ in the approximation, which means that $\mathbf{f}(\tau, \mathbf{x}(\tau))$ is approximated by interpolation and not extrapolation as the AB-schemes. By using $\mathbf{f}(t_n, \mathbf{x}_n)$ and $\mathbf{f}(t_{n+1}, \mathbf{x}_{n+1})$ to approximate $\mathbf{f}(\tau, \mathbf{x}(\tau))$ as a linear function in the interval (t_n, t_{n+1}) we find (see Figure 2):

AM1:

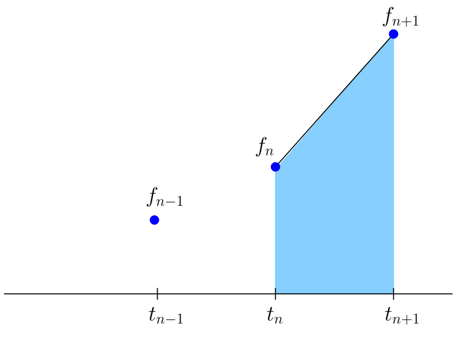
$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\left(\frac{1}{2}\mathbf{f}(t_{n+1}, \mathbf{x}_{n+1}) + \frac{1}{2}\mathbf{f}(t_n, \mathbf{x}_n)\right).$$

Similarly, by using $\mathbf{f}(t_{n-1}, \mathbf{x}_{n-1})$, $\mathbf{f}(t_n, \mathbf{x}_n)$ and $\mathbf{f}(t_{n+1}, \mathbf{x}_{n+1})$ to approximate $\mathbf{f}(\tau, \mathbf{x}(\tau))$ as a quadratic function in the interval (t_n, t_{n+1}) we find (see Figure 2):

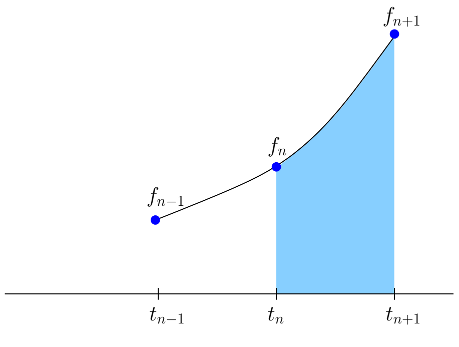
AM2:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\left(\frac{5}{12}\mathbf{f}(t_{n+1}, \mathbf{x}_{n+1}) + \frac{2}{3}\mathbf{f}(t_n, \mathbf{x}_n) - \frac{1}{12}\mathbf{f}(t_{n-1}, \mathbf{x}_{n-1})\right).$$

The global error for an Adams-Moulton method with s steps is $\mathcal{O}(h^{s+1})$.



(a) AM1



(b) AM2

Figure 2: Approximation of $\mathbf{f}(\tau, \mathbf{x}(\tau))$ for Adams-Moulton methods.