Overview of numerical methods for ODEs

In this lecture we provide a brief overview of the most common numerical methods to approximate the solution of an initial value problem for systems of ODEs of the form

$$\begin{cases} \dfrac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}, t) \\ \boldsymbol{y}(0) = \boldsymbol{y}_0 \end{cases} \tag{1}$$

where $\boldsymbol{y}(t) = [y_1(t) \cdots y_n(t)]^T$ is a (column) vector of phase variables, $\boldsymbol{f} : D \times [0, T] \to \mathbb{R}^n$, D is a subset of $\mathbb{R}^n$, and T is the integration period. Most of the material presented in this lecture can be found, e.g., in [2, 3, 4]. We assume that the initial value problem (1) well-posed, i.e., that it has a unique solution[1] (at least for some time $t\tau > 0$). We have seen that this is equivalent to assume that $\boldsymbol{f}(\boldsymbol{y}, t)$ is at least Lipschitz continuous in D and that $\boldsymbol{y}_0 \in D$.

The initial value problem (1) can be equivalently written as

$$\boldsymbol{y}(t) = \boldsymbol{y}(0) + \int_0^t \boldsymbol{f}(\boldsymbol{y}(s), s) ds. \tag{2}$$

i.e., as an integral equation for $\boldsymbol{y}(t)$.

Picard iteration method. The Picard iteration method is rarely used in practice to compute numerical solutions to ODEs, but rather to prove existence and uniqueness of solutions to ODEs. Picard's method is essentially a fixed point iteration in which the solution $\boldsymbol{y}(t)$ is approximated within a fixed time interval $[0, t]$ by a *sequence of functions*

$$\boldsymbol{y}^{[0]}(t) \to \boldsymbol{y}^{[1]}(t) \to \cdots \to \boldsymbol{y}^{[k]}(t) \to \cdots \tag{3}$$

generated by the fixed iteration rule

$$\boldsymbol{y}^{[k+1]}(t) = \boldsymbol{y}_0 + \int_0^t \boldsymbol{f}\left(\boldsymbol{y}^{[k]}(s), s\right) ds. \tag{4}$$

In (4) we can set $\boldsymbol{y}^{[0]}(t) = \boldsymbol{y}_0$ which, in principle, allows us to compute $\boldsymbol{y}^{[1]}(t)$. With $\boldsymbol{y}^{[1]}(t)$ available we can compute $\boldsymbol{y}^{[2]}(t)$, and so on and so forth. For the sequence of functions $\boldsymbol{y}^{[k]}(t)$ to converge from an arbitrary $\boldsymbol{y}^{[0]}(t)$, we need to make sure that the Frechét derivative of the nonlinear operator at the right hand side of (4) has a norm smaller than one, i.e., that the operator is a contraction. This translates to an upper bound for t, which depends on the Lipschitz constant of $\boldsymbol{f}$. The larger the Lipschitz constant, the smaller t. In other words, Picard iterations converge only if t is smaller than some t_{max} that depends on the Lipschitz constant of $\boldsymbol{f}$. Clearly, once $\boldsymbol{y}(t)$ has been computed to the desired accuracy we can use $y(t)$ as initial condition for the next sequence of Picard iterations, e.g., the sequence that yields the solution within the time interval $[t, 2t]$. The Picard iteration method is clearly not practical, as it involves the evaluation of a time integral at each iteration. Moreover, the function $y^{[k]}(t)$ that is being integrated is defined by an integral with t as one of the endpoints of the integral (see equation (4)).

Next, consider a partition of time interval $[0, T]$ into an evenly-spaced set of grid points:

$$t_0 = 0, \qquad t_N = T, \qquad t_{i+1} = t_i + \Delta t \qquad i = 0, \ldots, N-1, \tag{5}$$

[1] Of course, if the initial value problem (1) does not admit a unique solution then its numerical approximation might just pick one of the possible solutions.

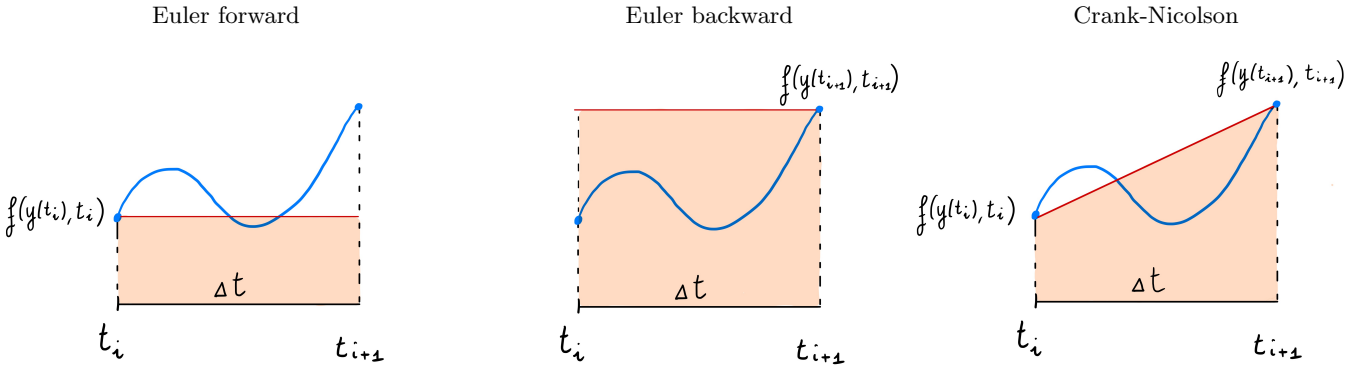

Figure 1: Approximations of the integral $\int_{t_i}^{t_{i+1}} \boldsymbol{f}(\boldsymbol{y}(s), s)ds$ in (6) leading to well-known numerical schemes.

Figure 2: Show how you integrate rectangle and trapezoidal rule to obtain the three methods (Euler forward/backward and CN - 3 Figures).

where $\Delta t = T/N$.

By using the semi-group property of (1) we can write (2) within each time interval $[t_k, t_{k+1}]$ as

$$\boldsymbol{y}(t_{k+1}) = \boldsymbol{y}(t_k) + \int_{t_k}^{t_{k+1}} \boldsymbol{f}\left(\boldsymbol{y}(s), s\right) ds. \tag{6}$$

This formulation is quite convenient for developing numerical methods based on *numerical quadrature formulas*, i.e., numerical approximations of the one-dimensional temporal integral appearing at the right hand side of (6).

Euler and Crank-Nicolson methods These are elementary methods obtained by approximating the integral at the right hand side of (6) by using the rectangle rule or the trapezoidal rule (see Figure 1). Specifically, consider the approximations

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}\left(\boldsymbol{y}(s), s\right) ds \simeq \Delta t \boldsymbol{f}\left(\boldsymbol{y}(t_k), t_k\right), \tag{7}$$

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}\left(\boldsymbol{y}(s), s\right) ds \simeq \Delta t \boldsymbol{f}\left(\boldsymbol{y}(t_{k+1}), t_{k+1}\right), \tag{8}$$

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}\left(\boldsymbol{y}(s), s\right) ds \simeq \frac{\Delta t}{2}\left(\boldsymbol{f}\left[\boldsymbol{y}(t_k), t_k\right) + \boldsymbol{f}\left(\boldsymbol{y}(t_{k+1}), t_{k+1}\right)\right]. \tag{9}$$

These quadrature rules yield, respectively, the following numerical schemes

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \Delta t \boldsymbol{f}\left(\boldsymbol{u}_k, t_k\right) \qquad \text{Euler forward method (explicit)}, \tag{10}$$

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \Delta t \boldsymbol{f}\left(\boldsymbol{u}_{k+1}, t_{k+1}\right) \qquad \text{Euler backward method (implicit)}, \tag{11}$$

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \frac{\Delta t}{2}\left[\boldsymbol{f}\left(\boldsymbol{u}_k, t_k\right) + \boldsymbol{f}\left(\boldsymbol{u}_{k+1}, t_{k+1}\right)\right] \qquad \text{Crank-Nicolson method (implicit)}. \tag{12}$$

In these methods $\boldsymbol{u}_k$ represents an approximation of the exact solution $\boldsymbol{y}(t_k)$, and we set $\boldsymbol{u}_0 = \boldsymbol{y}_0$ (initial condition). Both Euler and Crank-Nicolson methods are *one-step methods*. This means that the approximate solution at time t_{k+1}, i.e., $\boldsymbol{u}_{k+1}$, can be computed by knowing only the solution (or its approximation)

at time t_k. The Euler forward method allows us to compute $\boldsymbol{u}_{k+1}$ explicitly, given $\boldsymbol{u}_k$ and $\boldsymbol{f}(\boldsymbol{u}_k, t_k)$. On the other hand, the Euler backward and Crank-Nicolson methods are "implicit". This is because the approximate solution at time t_{k+1}, i.e., $\boldsymbol{u}_{k+1}$, cannot be computed explicitly based on $\boldsymbol{u}_k$, but it requires solving a nonlinear equation. Specifically, in the case of the Euler backward method we need to solve the nonlinear equation

$$\boldsymbol{u}_{k+1} = \boldsymbol{G}(\boldsymbol{u}_{k+1}) \qquad \text{where} \qquad \boldsymbol{G}(\boldsymbol{u}_{k+1}) = \boldsymbol{u}_k + \Delta t \boldsymbol{f}(\boldsymbol{u}_{k+1}, t_{k+1}). \tag{13}$$

Nonlinear equations of this form are essentially fixed point problems which can be solved numerically using iterative methods such as the Newton's method (provided $\boldsymbol{f}$ is of class C^1). Upon definition of

$$\boldsymbol{F}(\boldsymbol{u}_{k+1}) = \boldsymbol{u}_{k+1} - \boldsymbol{G}(\boldsymbol{u}_{k+1}) \tag{14}$$

we can equivalently write (13) as

$$\boldsymbol{F}(\boldsymbol{u}_{k+1}) = \boldsymbol{0}. \tag{15}$$

The Newton's method for nonlinear systems: The solution to the system of nonlinear equations (15) can be approximated by using the Newton's method or any other method for root finding. As is well known, the Newton's method generates a sequence of vectors $\boldsymbol{u}_{k+1}^{[j]}$ $(j = 0, 1, \ldots)$ converging to $\boldsymbol{u}_{k+1}$ under rather mild assumptions (see [4, Ch. 7]). The Newton's method can be formulated as[2]

$$\underbrace{\left[\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{G}}\left(\boldsymbol{u}_{k+1}^{[j]}\right)\right]}_{\text{matrix}} \underbrace{\left(\boldsymbol{u}_{k+1}^{[j+1]} - \boldsymbol{u}_{k+1}^{[j]}\right)}_{\text{vector}} = \underbrace{\boldsymbol{G}\left(\boldsymbol{u}_{k+1}^{[j]}\right) - \boldsymbol{u}_{k+1}^{[j]}}_{\text{vector}} \qquad j = 0, 1, \ldots, \tag{18}$$

where $\boldsymbol{J}_{\boldsymbol{G}}\left(\boldsymbol{u}_{k+1}^{[j]}\right)$ is the Jacobian matrix of $\boldsymbol{G}$ defined in equation (13), evaluated at $\boldsymbol{u}_{k+1}^{[j]}$. It is convenient to set the initial guess $\boldsymbol{u}_{k+1}^{[0]}$ for Newton's iteration as $\boldsymbol{u}_{k+1}^{[0]} = \boldsymbol{u}_k$, i.e., the numerical solution at previous time step. For Δt sufficiently small this guarantees that $\boldsymbol{u}_{k+1}^{[0]}$ is within the basin of attraction of $\boldsymbol{u}_{k+1}$. Note also that for Δt sufficiently small the matrix at the left hand side of (18) is a perturbation of the identity (the norm of $\boldsymbol{J}_{\boldsymbol{G}}$ goes to zero linearly in Δt), and therefore $\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{G}}$ is always invertible for sufficiently small Δt. Indeed $\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{G}}$ is a diagonally dominant matrix for small Δt. More rigorously, we have the following convergence result (see [4, Theorem 7.1]).

Theorem 1 (Convergence of Newton's method)**.** Let $\boldsymbol{F}(\boldsymbol{x})$ in equation (14) be of class C^1 in a convex open set $D \subseteq \mathbb{R}^n$ that contains a zero of $\boldsymbol{F}$, i.e., a point $\boldsymbol{x}^* \in D$ such that $\boldsymbol{F}(\boldsymbol{x}^*) = \boldsymbol{0}$. If $\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{x})$ is invertible at $\boldsymbol{x}^*$ (it always is for sufficiently small Δt), and $\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{x})$ is Lipschitz continuous in a neighborhood of $\boldsymbol{x}^*$, i.e.[3],

$$\|\boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{x}) - \boldsymbol{J}_{\boldsymbol{F}}(\boldsymbol{y})\| \leq L \|\boldsymbol{x} - \boldsymbol{y}\| \tag{19}$$

then there exists a neighborhood of $\boldsymbol{x}^*$ such that for any initial guess $\boldsymbol{x}^{[0]}$ in such a neighborhood we have that the sequence $\boldsymbol{x}^{[k]}$ generated by

$$\boldsymbol{J}_{\boldsymbol{F}}\left(\boldsymbol{x}^{[k]}\right)\left(\boldsymbol{x}^{[k+1]} - \boldsymbol{x}^{[k]}\right) = -\boldsymbol{F}\left(\boldsymbol{x}^{[k]}\right) \tag{20}$$

[2]Consider the Taylor series

$$\boldsymbol{F}\left(\boldsymbol{u}_{k+1}^{[j+1]}\right) = \boldsymbol{F}\left(\boldsymbol{u}_{k+1}^{[j]}\right) + \boldsymbol{J}_{\boldsymbol{F}}\left(\boldsymbol{u}_{k+1}^{[j]}\right)\left(\boldsymbol{u}_{k+1}^{[j+1]} - \boldsymbol{u}_{k+1}^{[j]}\right) + \cdots \qquad j = 0, 1, \ldots. \tag{16}$$

Setting $\boldsymbol{F}\left(\boldsymbol{u}_{k+1}^{[j+1]}\right) = \boldsymbol{0}$ yields

$$\boldsymbol{J}_{\boldsymbol{F}}\left(\boldsymbol{u}_{k+1}^{[j]}\right)\left(\boldsymbol{u}_{k+1}^{[j+1]} - \boldsymbol{u}_{k+1}^{[j]}\right) = -\boldsymbol{F}\left(\boldsymbol{u}_{k+1}^{[j]}\right) \qquad j = 0, 1, \ldots \tag{17}$$

which, upon substitution of (14), coincides with the Newton method (18).

[3]In equation (19) the matrix norm $\|\cdot\|$ at the left hand side is induced by the vector norm at the right hand side.

converges to $\boldsymbol{x}^*$ with order 2. In other words, there exists $m \in \mathbb{N}$ such that

$$\left\|\boldsymbol{x}^{[k+1]} - \boldsymbol{x}^*\right\| \leq C \left\|\boldsymbol{x}^{[k]} - \boldsymbol{x}^*\right\|^2 \quad \text{for all } k \geq m. \tag{21}$$

If we replace $\boldsymbol{u}_{k+1}$ at the right hand side of (12) with one step of the Euler forward scheme (10) we obtain the *Heun method*

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \frac{\Delta t}{2}\left[\boldsymbol{f}\left(\boldsymbol{u}_k, t_k\right) + \boldsymbol{f}\left(\boldsymbol{u}_k + \Delta t \boldsymbol{f}(\boldsymbol{u}_k, t_k), t_{k+1}\right)\right] \qquad \text{Heun method (explicit)} \tag{22}$$

The Heun method is a one-step explicit method that belongs to the class of two-stage explicit Runge-Kutta methods.

Remark: The Euler methods (10)-(11) can be derived also by replacing $d\boldsymbol{y}/dt$ in (1) with the first-order forward and backward finite-differentiation formulas

$$\frac{d\boldsymbol{y}(t_k)}{dt} \simeq \frac{\boldsymbol{y}(t_{k+1}) - \boldsymbol{y}(t_k)}{\Delta t} = \boldsymbol{f}(\boldsymbol{y}(t_k), t_k), \tag{23}$$

$$\frac{d\boldsymbol{y}(t_{k+1})}{dt} \simeq \frac{\boldsymbol{y}(t_{k+1}) - \boldsymbol{y}(t_k)}{\Delta t} = \boldsymbol{f}(\boldsymbol{y}(t_{k+1}), t_{k+1}). \tag{24}$$

The midpoint method. By approximating the integral at the right hand side of (6) using the midpoint rule yields

$$\int_{t_i}^{t_{i+1}} \boldsymbol{f}(\boldsymbol{y}(s), s) ds \simeq \Delta t \boldsymbol{f}\left(\boldsymbol{y}\left(t_i + \frac{\Delta t}{2}\right), t_i + \frac{\Delta t}{2}\right). \tag{25}$$

At this point, we approximate $\boldsymbol{y}(t_i + \Delta t/2)$ using the Euler forward method to obtain

$$\boldsymbol{y}\left(t_i + \frac{\Delta t}{2}\right) \simeq \boldsymbol{y}(t_i) + \frac{\Delta t}{2}\boldsymbol{f}(\boldsymbol{y}(t_i), t_i) \tag{26}$$

to obtain

$$\boldsymbol{u}_{i+1} = \boldsymbol{u}_i + \Delta t \boldsymbol{f}\left(\boldsymbol{u}_i + \frac{\Delta t}{2}\boldsymbol{f}(\boldsymbol{u}_i, t_i), t_i + \frac{\Delta t}{2}\right) \qquad \text{explicit mipoint method} \tag{27}$$

where $\boldsymbol{u}_i$ is an approximation of $\boldsymbol{y}(t_i)$. The explicit midpoint method is a one-step method that belongs to the class of two-stage explicit Runge-Kutta methods.

Approximating $\boldsymbol{y}(t_i + \Delta t/2)$ by the average

$$\boldsymbol{y}\left(t_i + \frac{\Delta t}{2}\right) \simeq \frac{\boldsymbol{y}(t_i) + \boldsymbol{y}(t_{i+1})}{2} \tag{28}$$

yields

$$\boldsymbol{u}_{i+1} = \boldsymbol{u}_i + \Delta t \boldsymbol{f}\left(\frac{\boldsymbol{u}_i + \boldsymbol{u}_{i+1}}{2}, t_i + \frac{\Delta t}{2}\right) \qquad \text{implicit mipoint method} \tag{29}$$

The implicit midpoint method is a *one-step symplectic integrator*, i.e., the scheme preserves the Hamiltonian when applied to Hamiltonian dynamical systems, e.g., pendulum or double-pendulum equations. There is a vast literature on *structure-preserving* integration methods for ordinary differential equations (see, e.g., [1] and the references therein).

Exercise: Solve the pendulum equations with the implicit midpoint method and the Euler forward method. Verify that the Hamiltonian is preserved in the case of the implicit midpoint method is used.

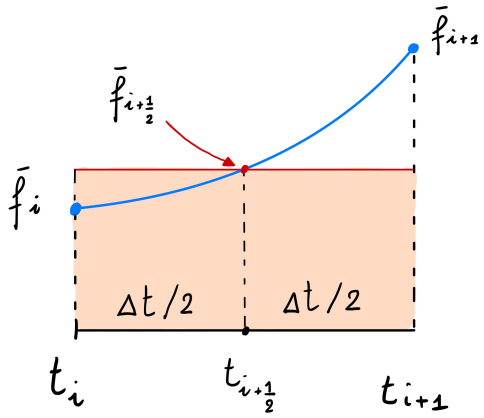

Figure 3: Approximation of the integral $\int_{t_i}^{t_{i+1}} \boldsymbol{f}(\boldsymbol{y}(s),s)ds$ in (6) leading to midpoint method. In this figure we set $\boldsymbol{f}_k = \boldsymbol{f}(\boldsymbol{y}(t_k),t_k)$.

Adams-Bashforth methods. These are *explicit multistep methods* constructed by replacing the integral at the right hand side of (6) with the integral of a polynomial interpolant of $\boldsymbol{f}(\boldsymbol{y}(s),s)$ at $\{t_k, t_{k-1}, \cdots t_{k-q}\}$ which is then extrapolated into $[t_k, t_{k+1}]$ to compute the integral. In other words, we introduce the following approximation

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}(\boldsymbol{y}(s),s)ds \simeq \int_{t_k}^{t_{k+1}} \Pi_q \boldsymbol{f}(\boldsymbol{y}(s),s)ds \tag{30}$$

where $\Pi_q \boldsymbol{f}(\boldsymbol{y}(s),s)$ is a polynomial of degree q interpolating

$$\{\boldsymbol{f}_k, \boldsymbol{f}_{k-1}, \ldots, \boldsymbol{f}_{k-q}\} \qquad \text{at} \qquad \{t_k, t_{k-1}, \ldots, t_{k-q}\} \tag{31}$$

where $\boldsymbol{f}_k = \boldsymbol{f}(\boldsymbol{y}(t_k),t_k)$. It is very convenient to use Lagrangian interpolation to derive the polynomial $\Pi_q \boldsymbol{f}$. Hereafter we derive the Adams-Bashforth (AB) methods for $q = 0, 1, 2$.

- *One-step Adams-Bashforth method (AB1)*:

$$q = 0 \quad \Rightarrow \quad \Pi_0 \boldsymbol{f}(\boldsymbol{y}(s),s) = \boldsymbol{f}_k \tag{32}$$

  where $\boldsymbol{f}_k = \boldsymbol{f}(\boldsymbol{y}(t_k),t_k)$. Hence,

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}(\boldsymbol{y}(s),s)ds \simeq \int_{t_k}^{t_{k+1}} \Pi_0 \boldsymbol{f}(\boldsymbol{y}(s),s)ds = \Delta t \boldsymbol{f}(\boldsymbol{y}(t_k),t_k). \tag{33}$$

  This yields

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \Delta t \boldsymbol{f}(\boldsymbol{u}_k,t_k) \qquad \text{AB1 method.} \tag{34}$$

  Note that the AB1 method coincides with the Euler forward method.

- *Two-step Adams-Bashforth method (AB2)*:

$$q = 1 \quad \Rightarrow \quad \Pi_1 \boldsymbol{f}(\boldsymbol{y}(s),s) = \boldsymbol{f}_k l_k(s) + \boldsymbol{f}_{k-1} l_{k-1}(s) \tag{35}$$

  where $l_k(s)$ and $l_{k-1}(s)$ are Lagrange characteristic polynomials

$$l_k(s) = \frac{s - t_{k-1}}{t_k - t_{k-1}}, \qquad l_{k-1}(s) = \frac{s - t_k}{t_{k-1} - t_k}. \tag{36}$$

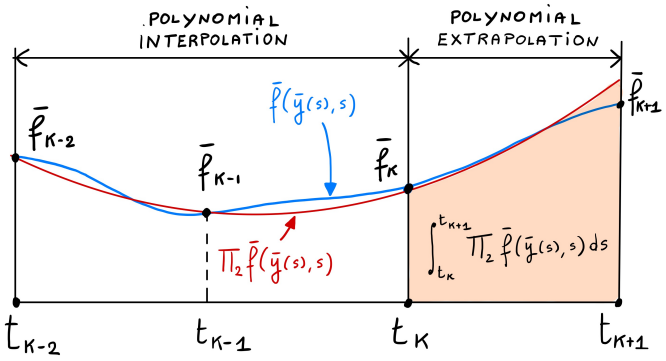

Figure 4: Derivation of the three-step Adams-Bashforth scheme (AB3). We first construct the polynomial $\Pi_2\boldsymbol{f}$ that interpolates $\boldsymbol{f}(\boldsymbol{y}(s),s)$ at t_{k-2}, t_{k-1} and t_k. Subsequently, we extrapolate $\Pi_2\boldsymbol{f}$ to $[t_k,t_{k+1}]$ so that we can compute the integral in equation (30).

This yields

$$\Pi_1\boldsymbol{f}(\boldsymbol{y}(s),s)=\boldsymbol{f}_k\frac{s-t_{k-1}}{t_k-t_{k-1}}+\boldsymbol{f}_{k-1}\frac{s-t_k}{t_{k-1}-t_k}. \tag{37}$$

The (linear) polynomial $\Pi_1\boldsymbol{f}(\boldsymbol{y}(s),s)$ is constructed in $[t_{k-1},t_k]$ and it can be integrated exactly in $[t_k,t_{k+1}]$ using the trapezoidal rule. To this end, we notice that

$$\Pi_1\boldsymbol{f}(\boldsymbol{y}(t_k),t_k)=\boldsymbol{f}_k, \tag{38}$$

$$\begin{aligned}\Pi_1\boldsymbol{f}(\boldsymbol{y}(t_{k+1}),t_{k+1})=&\boldsymbol{f}_k\frac{t_{k+1}-t_{k-1}}{t_k-t_{k-1}}+\boldsymbol{f}_{k-1}\frac{t_{k+1}-t_k}{t_{k-1}-t_k}\\ =&2\boldsymbol{f}_k-\boldsymbol{f}_{k-1}.\end{aligned} \tag{39}$$

which gives us

$$\int_{t_k}^{t_{k+1}}\boldsymbol{f}(\boldsymbol{y}(s),s)ds\simeq\int_{t_k}^{t_{k+1}}\Pi_1\boldsymbol{f}(\boldsymbol{y}(s),s)ds=\frac{\Delta t}{2}\left(3\boldsymbol{f}_k-\boldsymbol{f}_{k-1}\right). \tag{40}$$

Substituting this back into (6) yields the scheme

$$\boldsymbol{u}_{k+1}=\boldsymbol{u}_k+\frac{\Delta t}{2}\left[3\boldsymbol{f}(\boldsymbol{u}_k,t_k)-\boldsymbol{f}(\boldsymbol{u}_{k-1},t_{k-1})\right]\qquad\text{AB2 method} \tag{41}$$

- *Three-step Adams-Bashforth method (AB3)*: With reference to Figure (4)

$$q=3\quad\Rightarrow\quad\Pi_2\boldsymbol{f}(\boldsymbol{y}(s),s)=\boldsymbol{f}_k l_k(s)+\boldsymbol{f}_{k-1}l_{k-1}(s)+\boldsymbol{f}_{k-2}l_{k-2}(s) \tag{42}$$

where $l_k(s)$, $l_{k-1}(s)$ and $l_{k-2}(s)$ are Lagrange characteristic polynomials

$$l_k(s)=\frac{s-t_{k-1}}{t_k-t_{k-1}}\frac{s-t_{k-2}}{t_k-t_{k-2}}, \tag{43}$$

$$l_{k-1}(s)=\frac{s-t_k}{t_{k-1}-t_k}\frac{s-t_{k-2}}{t_{k-1}-t_{k-2}}, \tag{44}$$

$$l_{k-2}(s)=\frac{s-t_k}{t_{k-2}-t_k}\frac{s-t_{k-1}}{t_{k-2}-t_{k-1}}. \tag{45}$$

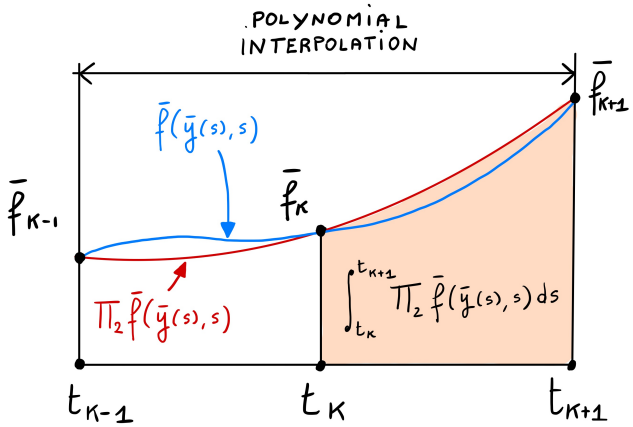

Figure 5: Derivation of the two-step Adams-Moulton scheme (AM2). We first construct the polynomial $\Pi_2\boldsymbol{f}$ that interpolates $\boldsymbol{f}(\boldsymbol{y}(s),s)$ at t_{k-1}, t_k and t_{k+1}. Subsequently, we use $\Pi_2\boldsymbol{f}$ polynomial to approximate the integral in equation (30).

By integrating $\Pi_2\boldsymbol{f}(\boldsymbol{y}(s),s)$ from t_k to t_{k+1} we obtain

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}(\boldsymbol{y}(s),s)ds \simeq \int_{t_k}^{t_{k+1}} \Pi_2\boldsymbol{f}(\boldsymbol{y}(s),s)ds = \frac{\Delta t}{12}\left(23\boldsymbol{f}_k - 16\boldsymbol{f}_{k-1} + 5\boldsymbol{f}_{k-2}\right). \tag{46}$$

Substituting this back into (6) yields the scheme

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \frac{\Delta t}{12}\left[23\boldsymbol{f}(\boldsymbol{u}_k,t_k) - 16\boldsymbol{f}(\boldsymbol{u}_{k-1},t_{k-1}) + 5\boldsymbol{f}(\boldsymbol{u}_{k-2},t_{k-2})\right] \qquad \text{AB3 method} \tag{47}$$

Higher-order Adams-Bashforth schemes can be obtained similarly. Note that in order to start-up a linear multistep scheme we need to compute the solution at the intermediate steps using different methods. For example, we could start-up the AB2 method with one step of the Heun method (to compute $\boldsymbol{u}_1$) and then carry on the integration using the scheme (41)

Adams-Moulton methods. These are implicit multistep methods in which the time integral at the right hand-side of (6) is approximated by replacing $\boldsymbol{f}(\boldsymbol{y}(s),s)$ by a polynomial $\Pi_q\boldsymbol{f}(\boldsymbol{y}(s),s)$ of degree q interpolating

$$\{\boldsymbol{f}_{k+1},\boldsymbol{f}_k,\ldots,\boldsymbol{f}_{k-q+1}\} \qquad \text{at} \qquad \{t_{k+1},t_{k-1},\ldots,t_{k-q+1}\} \tag{48}$$

The main difference with respect to the Adams-Bashforth method is that there is no extrapolation step, i.e., the point $(t_{k+1},\boldsymbol{f}_{k+1})$ is included in the interpolation (see Figure 5). Let us derive the Adams-Moulton schemes for $q=0,1,2$.

- *One-step Adams-Moulton method (AM0)*:

$$q=0 \quad \Rightarrow \quad \Pi_0\boldsymbol{f}(\boldsymbol{y}(s),s) = \boldsymbol{f}_{k+1} \tag{49}$$

where $\boldsymbol{f}_{k+1} = \boldsymbol{f}(\boldsymbol{y}(t_{k+1}),t_{k+1})$. Hence,

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}(\boldsymbol{y}(s),s)ds \simeq \int_{t_k}^{t_{k+1}} \Pi_0\boldsymbol{f}(\boldsymbol{y}(s),s)ds = \Delta t\boldsymbol{f}(\boldsymbol{y}(t_{k+1}),t_{k+1}). \tag{50}$$

This yields

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \Delta t\boldsymbol{f}(\boldsymbol{u}_{k+1},t_{k+1}) \qquad \text{AM0 method.} \tag{51}$$

Note that the AM0 method coincides with the Euler backward method.

- *One-step Adams-Moulton method (AM1)*:

$$q = 1 \quad \Rightarrow \quad \Pi_1 \boldsymbol{f}(\boldsymbol{y}(s), s) = \boldsymbol{f}_{k+1} l_{k+1}(s) + \boldsymbol{f}_k l_k(s) \tag{52}$$

where $l_k(s)$ and $l_{k-1}(s)$ are Lagrange characteristic polynomials

$$l_{k+1}(s) = \frac{s - t_k}{t_{k+1} - t_k}, \qquad l_k(s) = \frac{s - t_{k+1}}{t_k - t_{k+1}}. \tag{53}$$

This yields

$$\Pi_1 \boldsymbol{f}(\boldsymbol{y}(s), s) = \boldsymbol{f}_{k+1} \frac{s - t_k}{t_{k+1} - t_k} + \boldsymbol{f}_k \frac{s - t_{k+1}}{t_k - t_{k+1}}. \tag{54}$$

The (linear) polynomial $\Pi_1 \boldsymbol{f}(\boldsymbol{y}(s), s)$ is constructed in $[t_k, t_{k+1}]$ and it can be integrated exactly in the same interval. To this end, we notice that

$$\Pi_1 \boldsymbol{f}(\boldsymbol{y}(t_k), t_k) = \boldsymbol{f}_k, \tag{55}$$

$$\Pi_1 \boldsymbol{f}(\boldsymbol{y}(t_{k+1}), t_{k+1}) = \boldsymbol{f}_{k+1} \tag{56}$$

which imply

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}(\boldsymbol{y}(s), s) ds \simeq \int_{t_k}^{t_{k+1}} \Pi_1 \boldsymbol{f}(\boldsymbol{y}(s), s) ds = \frac{\Delta t}{2} \left(\boldsymbol{f}_{k+1} + \boldsymbol{f}_k \right). \tag{57}$$

Substituting this back into (6) yields the scheme

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \frac{\Delta t}{2} \left[\boldsymbol{f}(\boldsymbol{u}_{k+1}, t_{k+1}) + \boldsymbol{f}(\boldsymbol{u}_k, t_k) \right] \qquad \text{AM1 method} \tag{58}$$

Hence, the AM1 scheme coincides with the Crank-Nicolson scheme.

- *Two-step Adams-Moulton method (AM2)*:

$$q = 2 \quad \Rightarrow \quad \Pi_2 \boldsymbol{f}(\boldsymbol{y}(s), s) = \boldsymbol{f}_{k+1} l_{k+1}(s) + \boldsymbol{f}_k l_k(s) + \boldsymbol{f}_{k-1} l_{k-1}(s). \tag{59}$$

where $l_{k+1}(s)$, $l_k(s)$ and $l_{k-1}(s)$ are Lagrange characteristic polynomials

$$l_{k+1}(s) = \frac{s - t_k}{t_{k+1} - t_k} \frac{s - t_{k-1}}{t_{k+1} - t_{k-1}}, \tag{60}$$

$$l_k(s) = \frac{s - t_{k+1}}{t_k - t_{k+1}} \frac{s - t_{k-1}}{t_k - t_{k-1}}, \tag{61}$$

$$l_{k-1}(s) = \frac{s - t_{k+1}}{t_{k-1} - t_{k+1}} \frac{s - t_k}{t_{k-1} - t_k}. \tag{62}$$

By integrating $\Pi_2 \boldsymbol{f}(\boldsymbol{y}(s), s)$ from t_k to t_{k+1} we obtain

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}(\boldsymbol{y}(s), s) ds \simeq \int_{t_k}^{t_{k+1}} \Pi_2 \boldsymbol{f}(\boldsymbol{y}(s), s) ds = \frac{\Delta t}{12} \left(5\boldsymbol{f}_{k+1} + 8\boldsymbol{f}_k - \boldsymbol{f}_{k-1} \right). \tag{63}$$

Substituting this back into (6) yields the scheme

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \frac{\Delta t}{12} \left[5\boldsymbol{f}(\boldsymbol{u}_{k+1}, t_{k+1}) + 8\boldsymbol{f}(\boldsymbol{u}_k, t_k) - \boldsymbol{f}(\boldsymbol{u}_{k-1}, t_{k-1}) \right] \qquad \text{AM2 method.} \tag{64}$$

Backward differentiation formulas (BDF) methods. These are linear implicit multistep methods that perform well for stiff problems. These methods are obtained by approximating $d\boldsymbol{y}(t)/dt$ in (1) using a backward finite-difference formula. These formulas are obtained by interpolating $\boldsymbol{y}(s)$ at $\{t_{k+1}, t_k, \ldots, t_{k-q+1}\}$ with a polynomial of degree q, differentiating the polynomial and evaluating the derivative at $t = t_{k+1}$ (see Figure 6). Let us derive the first few BDF methods.

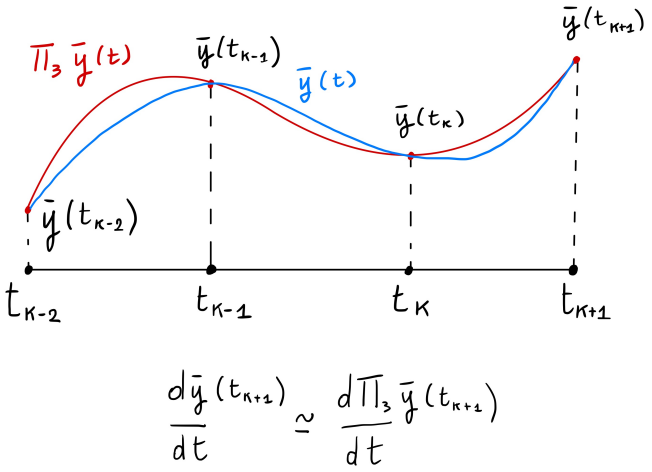

Figure 6: Derivation of the three-step backward differentiation formula (BDF3) method. We first construct the polynomial $\Pi_3 \boldsymbol{f}$ that interpolates $\boldsymbol{y}(t)$ at t_{k-2}, t_{k-1}, t_k, and t_{k+1}. Subsequently, approximate the derivative of $\boldsymbol{y}(t)$ at t_{k+1} with the derivative of the polynomial $\Pi_3 \boldsymbol{y}(t)$ at t_{k+1}.

- *One-step BDF method (BDF1):*

$$\Pi_1 \boldsymbol{y}(s) = \boldsymbol{y}_{k+1} l_{k+1}(s) + \boldsymbol{y}_k l_k(s) \tag{65}$$

where the Lagrange characteristic polynomials are given by

$$l_{k+1}(s) = \frac{s - t_k}{t_{k+1} - t_k}, \qquad l_k(s) = \frac{s - t_{k+1}}{t_k - t_{k+1}} \tag{66}$$

We approximate the derivative of $\boldsymbol{y}(s)$ at t_{k+1} with the derivative of the polynomial $\Pi_1 \boldsymbol{y}(s)$ at t_{k+1}, i.e.,

$$\frac{d\boldsymbol{y}(t_{k+1})}{dt} \simeq \frac{d\Pi_1 \boldsymbol{y}(t_{k+1})}{dt} = \frac{\boldsymbol{y}_{k+1} - \boldsymbol{y}_k}{\Delta t}. \tag{67}$$

Substituting this approximation into the exact equation

$$\frac{d\boldsymbol{y}(t_{k+1})}{dt} = \boldsymbol{f}(\boldsymbol{y}(t_{k+1}), t_{k+1}) \tag{68}$$

yields the scheme

$$\boldsymbol{u}_{k+1} - \boldsymbol{u}_k = \Delta t \boldsymbol{f}(\boldsymbol{u}_{k+1}, t_{k+1}). \tag{69}$$

Note that BDF1 coincides with the Euler backward scheme.

- *Two-step BDF method (BDF2):*

$$\Pi_2 \boldsymbol{y}(s) = \boldsymbol{y}_{k+1} l_{k+1}(s) + \boldsymbol{y}_k l_k(s) + \boldsymbol{y}_{k-1} l_{k-1}(s) \tag{70}$$

where the Lagrange characteristic polynomials are given by

$$l_{k+1}(s) = \frac{s - t_k}{t_{k+1} - t_k} \frac{s - t_{k-1}}{t_{k+1} - t_{k-1}}, \tag{71}$$

$$l_k(s) = \frac{s - t_{k+1}}{t_k - t_{k+1}} \frac{s - t_{k-1}}{t_k - t_{k-1}}, \tag{72}$$

$$l_{k-1}(s) = \frac{s - t_{k+1}}{t_{k-1} - t_{k+1}} \frac{s - t_k}{t_{k-1} - t_k}. \tag{73}$$

We approximate the derivative of $\boldsymbol{y}(s)$ at t_{k+1} with the derivative of the polynomial $\Pi_1\boldsymbol{y}(s)$ at t_{k+1}, i.e.,

$$\frac{d\boldsymbol{y}(t_{k+1})}{dt} \simeq \frac{d\Pi_2\boldsymbol{y}(t_{k+1})}{dt} = \frac{3\boldsymbol{y}_{k+1} - 4\boldsymbol{y}_k + \boldsymbol{y}_{k-1}}{2\Delta t}. \tag{74}$$

Substituting this approximation into the exact equation

$$\frac{d\boldsymbol{y}(t_{k+1})}{dt} = \boldsymbol{f}(\boldsymbol{y}(t_{k+1}), t_{k+1}) \tag{75}$$

yields the scheme

$$\frac{3}{2}\boldsymbol{u}_{k+1} - 2\boldsymbol{u}_k - \frac{1}{2}\boldsymbol{u}_{k-1} = \Delta t\boldsymbol{f}(\boldsymbol{u}_{k+1}, t_{k+1}). \tag{76}$$

- *Three-step BDF method (BDF3):* By following a similar procedure as in BDF2 it is straightforward to show that

$$\frac{11}{6}\boldsymbol{u}_{k+1} - 3\boldsymbol{u}_k + \frac{3}{2}\boldsymbol{u}_{k-1} - \frac{1}{3}\boldsymbol{u}_{k-2} = \Delta t\boldsymbol{f}(\boldsymbol{u}_{k+1}, t_{k+1}). \tag{77}$$

General form of a linear multistep method (LMM). The general form of a linear multistep method is

$$\sum_{j=0}^{q} \alpha_j \boldsymbol{u}_{k+j} = \Delta t \sum_{j=0}^{q} \beta_j \underbrace{\boldsymbol{f}(\boldsymbol{u}_{k+j}, t_{k+j})}_{\boldsymbol{f}_{k+j}}. \tag{78}$$

Note that Adams-Bashforth, Adams-Moulton and BDF methods are all in the form (78). To avoid non-uniqueness of coefficients due to rescaling we set $\alpha_q = 1$. Clearly, if $\beta_q = 0$ and then the method is *explicit*. On the other hand, if $\beta_q \neq 0$ then the method is *implicit*. Let us provide a few examples.

Example: The AB3 method

$$\boldsymbol{u}_{k+3} = \boldsymbol{u}_{k+2} + \frac{\Delta t}{12}\left(23\boldsymbol{f}_{k+2} - 16\boldsymbol{f}_{k+1} + 5\boldsymbol{f}_k\right) \tag{79}$$

can be written in the form (78) by setting

$$\alpha_3 = 1 \quad \alpha_2 = -1, \quad \alpha_1 = 0, \qquad \alpha_0 = 0,$$

$$\beta_3 = 0, \quad \beta_2 = \frac{23}{12}, \quad \beta_1 = -\frac{16}{12}, \quad \beta_0 = \frac{5}{12}.$$

Note that $(\alpha_3, \beta_3) = (1, 0)$ (the method is explicit) and

$$\sum_{j=0}^{3} \beta_j = 1. \tag{80}$$

Example: The BDF2 method

$$\boldsymbol{u}_{k+2} - \frac{4}{3}\boldsymbol{u}_{k+1} + \frac{1}{3}\boldsymbol{u}_k = \frac{2}{3}\Delta t\boldsymbol{f}_{k+2} \tag{81}$$

can be written in the form (78) by setting

$$\alpha_2 = 1, \qquad \alpha_1 = -\frac{4}{3}, \qquad \alpha_0 = \frac{1}{3}, \qquad \beta_2 = \frac{2}{3}, \qquad \beta_1 = 0, \qquad \beta_0 = 0. \tag{82}$$

Note that $(\alpha_2, \beta_2) = (1, 1/3)$ (the method is implicit) and

$$\sum_{j=0}^{2} \alpha_j = 0, \qquad \sum_{j=0}^{2} (\beta_j - j\alpha_j) = 0. \tag{83}$$

As we will see the conditions (80) and (83) guarantee that AB3 and BDF2 are *consistent* methods, i.e., that the truncation error of these methods goes to zero as we send Δt to zero. Roughly speaking this means that the numerical schemes (79) and (81) converge to the ODE (1) as $\Delta t \to 0$. This is necessary but not sufficient for the numerical solution generated by the scheme to converge to the analytical solution. The other element that is needed for convergence is *zero-stability*. The consistency conditions for a general linear q-step method are

$$\sum_{j=0}^{q} \alpha_j = 0, \qquad \sum_{j=0}^{q} (\beta_j - j\alpha_j) = 0. \tag{84}$$

Remark: All linear multistep methods rely on a polynomial interpolation process on an evenly-spaced temporal grid. As is well-known, polynomial interpolation on evenly spaced grids is, in general, ill-conditioned and can undergo a severe Gibbs phenomenon depending on the function. However, the process of interpolating a function with a polynomial of degree $q+1$ within a very small a time interval (equal to $q\Delta t$ for a q-step method) is not ill-conditioned. The reason can be traced back to the fact that we are interpolating on a small time interval $[t - q\Delta t, t]$ in which the function behaves more or less almost like a line. More rigorously, if $g(t)$ is any function of class C^{q+1} in $[t - q\Delta t, t]$ and $\Pi_q g(t)$ is a polynomial of degree q that interpolates $g(t)$ at $\{t, t - \Delta t, \ldots, t - q\Delta t\}$ then we have the error estimate

$$|g(t) - \Pi_q g(t)| = \frac{1}{(q+1)!} \left| \frac{d^{q+1} f(\xi)}{dt^{q+1}} \right| \prod_{j=0}^{q} |(t - t_j)| \leq \frac{(q\Delta t)^{q+1}}{(q+1)!} \left| \frac{d^{q+1} f(\xi)}{dt^{q+1}} \right| \qquad \xi \in [t - q\Delta t, t], \tag{85}$$

which clearly goes to zero for sufficiently small Δt.

Runge-Kutta methods. Runge-Kutta (RK) methods are one-step methods (implicit or explicit) that aim at increasing accuracy by increasing the number of function evaluations within each time step. The general form of a RK method with s stages is

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \Delta t \sum_{i=1}^{s} b_i \boldsymbol{K}_i \tag{86}$$

where

$$\boldsymbol{K}_i = \boldsymbol{f} \left(\boldsymbol{u}_k + \Delta t \sum_{j=1}^{s} a_{ij} \boldsymbol{K}_j, t_k + c_i \Delta t \right). \tag{87}$$

The coefficients of the RK method are usually collected in a table called *Butcher array*

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

The elements a_{ij} in the Butcher table can be positive or negative. Constency of RK methods, i.e., the fact that (86) converge to (1) as $\Delta t \to 0$ implies the following conditions

$$\sum_{j=1}^{s} b_i = 1. \tag{88}$$

Moreover, we assume that

$$c_i = \sum_{j=1}^{s} a_{ij}. \tag{89}$$

Such a "row-sum condition" is not needed for a consistent method.

If $a_{ij} = 0$ for $i \geq j$ then each $\boldsymbol{K}_i$ can be computed recursively from the previous ones and the RK method is *explicit*. Otherwise the RK method is implicit. Let us provide a few examples.

- *Euler forward method*: The Euler forward method can be seen as a one-step explicit RK method. The Butcher array corresponding to such explicit RK1 method is

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

- *Heun method*: The Heun mehod (22) is a two-stage explicit Runge-Kutta method. The Butcher array for the Heun method is:

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

  This table corresponds to the following RK2 method

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \frac{\Delta t}{2} \left(\boldsymbol{K}_1 + \boldsymbol{K}_2 \right), \tag{90}$$

  where

$$\boldsymbol{K}_1 = \boldsymbol{f} \left(\boldsymbol{u}_k, t_k \right), \tag{91}$$
$$\boldsymbol{K}_2 = \boldsymbol{f} \left(\boldsymbol{u}_k + \Delta t \boldsymbol{f}(\boldsymbol{u}_k, t_k), t_k + \Delta t \right). \tag{92}$$

- *Crank-Nicolson method*: The Crank-Nicolson (CN) mehod (12) is a two-stage implicit Runge-Kutta method. The Butcher array corresponding to such method is:

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

  This table corresponds to the following RK2 method

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \frac{\Delta t}{2} \left(\boldsymbol{K}_1 + \boldsymbol{K}_2 \right), \tag{93}$$

  where

$$\boldsymbol{K}_1 = \boldsymbol{f} \left(\boldsymbol{u}_k, t_k \right), \tag{94}$$
$$\boldsymbol{K}_2 = \boldsymbol{f} \left(\boldsymbol{u}_k + \frac{\Delta t}{2} \left(\boldsymbol{K}_1 + \boldsymbol{K}_2 \right), t_{k+1} \right). \tag{95}$$

From equation (93) we see that

$$\frac{\Delta t}{2}\left(\boldsymbol{K}_1+\boldsymbol{K}_2\right)=\boldsymbol{u}_{k+1}-\boldsymbol{u}_k. \tag{96}$$

Substituting this expression into (95) yields

$$\boldsymbol{K}_1=\boldsymbol{f}\left(\boldsymbol{u}_k,t_k\right),\qquad \boldsymbol{K}_2=\boldsymbol{f}\left(\boldsymbol{u}_{k+1},t_{k+1}\right). \tag{97}$$

- *Kutta's method*: This method is an explicit 3-stage method corresponding to the Butcher array:

0	0	0	0
1/2	1/2	0	0
1	−1	2	0
	1/6	2/3	1/6

The method can be written as

$$\boldsymbol{K}_1=\boldsymbol{f}(\boldsymbol{u}_k,t_k), \tag{98}$$
$$\boldsymbol{K}_2=\boldsymbol{f}\left(\boldsymbol{u}_k+\frac{\Delta t}{2}\boldsymbol{K}_1,t_k+\frac{\Delta t}{2}\right), \tag{99}$$
$$\boldsymbol{K}_3=\boldsymbol{f}\left(\boldsymbol{u}_k-\Delta t\boldsymbol{K}_1+2\Delta t\boldsymbol{K}_2,t_k+\Delta t\right), \tag{100}$$

$$\boldsymbol{u}_{k+1}=\boldsymbol{u}_k+\frac{\Delta t}{6}\left(\boldsymbol{K}_1+4\boldsymbol{K}_2+\boldsymbol{K}_3\right). \tag{101}$$

- *Runge-Kutta method (RK4)*: The most famous RK method is perhaps the one proposed in the original paper by Runge and Kutta. Such a method is an explicit 4-stage method corresponding to the Butcher array:

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	1/3	1/3	1/6

The method can be written as

$$\boldsymbol{K}_1=\boldsymbol{f}(\boldsymbol{u}_k,t_k), \tag{102}$$
$$\boldsymbol{K}_2=\boldsymbol{f}\left(\boldsymbol{u}_k+\frac{\Delta t}{2}\boldsymbol{K}_1,t_k+\frac{\Delta t}{2}\right), \tag{103}$$
$$\boldsymbol{K}_3=\boldsymbol{f}\left(\boldsymbol{u}_k+\frac{\Delta t}{2}\boldsymbol{K}_2,t_k+\frac{\Delta t}{2}\right), \tag{104}$$
$$\boldsymbol{K}_4=\boldsymbol{f}\left(\boldsymbol{u}_k+\Delta t\boldsymbol{K}_3,t_k+\Delta t\right), \tag{105}$$

$$\boldsymbol{u}_{k+1}=\boldsymbol{u}_k+\frac{\Delta t}{6}\left(\boldsymbol{K}_1+2\boldsymbol{K}_2+2\boldsymbol{K}_3+\boldsymbol{K}_4\right). \tag{106}$$

$$\begin{array}{c|cc} 0 & 0 & 0 \\ c & c & 0 \\ \hline & b_1 & b_2 \end{array}$$

Derivation of explicit RK2 methods. Let us show how to derive an arbitrary explicit two-stage RK method. To this end we first write the Butcher array:

where $b_1 + b_2 = 1$. The RK2 method corresponding to this table can be written explicitly as

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \Delta t(b_1 \boldsymbol{K}_1 + b_2 \boldsymbol{K}_2) \tag{107}$$

where

$$\boldsymbol{K}_1 = \boldsymbol{f}(\boldsymbol{u}_k, t_k) \tag{108}$$
$$\boldsymbol{K}_2 = \boldsymbol{f}(\boldsymbol{u}_k + c\Delta t \boldsymbol{K}_1, t_k + c\Delta t). \tag{109}$$

Expand $\boldsymbol{K}_2$ in a Taylor series in Δt to obtain

$$\boldsymbol{K}_2 = \boldsymbol{K}_1 + c\Delta t \left(\sum_{j=1}^{n} \frac{\partial \boldsymbol{f}}{\partial y_j} K_{1j} + \frac{\partial \boldsymbol{f}}{\partial t} \right) + \cdots . \tag{110}$$

A substitution of this expression into (107) yields

$$\boldsymbol{u}_{k+1} = \boldsymbol{u}_k + \Delta t(b_1 + b_2)\boldsymbol{f}(\boldsymbol{u}_k, t_k) + b_2 c(\Delta t)^2 \left(\sum_{j=1}^{n} \frac{\partial \boldsymbol{f}(\boldsymbol{u}_k, t_k)}{\partial y_j} f_j(\boldsymbol{u}_k, t_k) + \frac{\partial \boldsymbol{f}(\boldsymbol{u}_k, t_k)}{\partial t} \right). \tag{111}$$

Next, we expand the solution to the ODE (1) in a Taylor series at time $t_{k+1} = t_k + \Delta t)$, assuming that $\boldsymbol{y}(t_k) = \boldsymbol{u}_k$. This yields

$$\boldsymbol{y}(t_{k+1}) = \boldsymbol{u}(t_k) + \frac{d\boldsymbol{y}(t_k)}{dt}\Delta t + \frac{(\Delta t)^2}{2}\frac{d^2\boldsymbol{y}(t_k)}{dt^2} + \cdots . \tag{112}$$

By using (1) and the chain rule we obtain:

$$\frac{d\boldsymbol{y}(t_k)}{dt} = \boldsymbol{f}(\boldsymbol{y}_k, t_k), \qquad \frac{d^2\boldsymbol{y}(t_k)}{dt^2} = \sum_{j=1}^{n} \frac{\partial \boldsymbol{f}(\boldsymbol{y}_k, t_k)}{\partial y_j} f_j(\boldsymbol{y}_k, t_k) + \frac{\partial \boldsymbol{f}(\boldsymbol{y}_k, t_k)}{\partial t}. \tag{113}$$

Assuming that $\boldsymbol{y}_k = \boldsymbol{u}_k$ and matching the terms multiplying the same powers of Δt in (111) and (112) we obtain[4]

$$b_1 + b_2 = 1 \qquad \text{and} \qquad cb_2 = \frac{1}{2}. \tag{114}$$

This is a system of 2 equations in 3 unknowns. Thus, there is an *infinite number* of explicit (and consistent) RK2 methods. For example, setting

$$c = 1, \qquad b_1 = b_2 = \frac{1}{2} \qquad \text{yields the } \textit{Heun method } (22). \tag{115}$$

On the other hand, setting

$$c = \frac{1}{2}, \quad b_1 = 0 \quad b_2 = 1 \qquad \text{yields the } \textit{explicit midpoint method } (27). \tag{116}$$

[4]The condition $b_1 + b_2 = 1$ is a consistency condition which guarantees that the scheme (107) converges to the ODE (1) in the limit $\Delta t \to 0$.

By using the Taylor series approach discussed in this section, it is possible to derive conditions on the entries of the Butcher array for RK methods with an arbitrary number of stages. Essentially, we perform a Taylor series of the RK method in Δt and then match it with the Taylor series expansion of the solution up to a given order. The corresponding equations for the coefficients, e.g., (114) are called "stage-order" conditions. This is discussed, e.g., in [2, §5.9]. For example, it can be shown that for a three stage RK method

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ c_2 & a_{21} & 0 & 0 \\ c_3 & a_{31} & a_{32} & 0 \\ \hline & b_1 & b_2 & b_3 \end{array}$$

we obtain the order conditions

$$\begin{cases} b_1 + b_2 + b_3 = 1 \\ b_2 c_2 + b_3 c_3 = \dfrac{1}{2} \\ b_2 c_2^2 + b_3 c_3^2 = \dfrac{1}{3} \\ b_3 a_{32} c_2 = \dfrac{1}{6} \end{cases} \tag{117}$$

which yields to one two-parameter family of solutions and two one-parameter families of solutions [2, p. 178]. As easily seen, the Taylor series approach rapidly become intractable as the number of stages increases, and so does the corresponding set of order conditions. Fortunately, there is a more efficient way to derive conditions such as (114) by using rooted trees (see [2, §5.6]).

Derivation of implicit RK methods. Implicit RK methods can be derived from the integral formulation (6) of the Cauchy problem. In fact, if a Gauss quadrature formula with s nodes in $[tk, t_{k+1}]$ is employed to approximate the integral at the right hand side of (6), we obtain

$$\int_{t_k}^{t_{k+1}} \boldsymbol{f}(\boldsymbol{y}(s), s) ds \simeq \Delta t \sum_{j=1}^{s} b_j \boldsymbol{f}\left(\boldsymbol{y}(t_k + c_j \Delta t), t_k + c_j \Delta t\right). \tag{118}$$

Here, we denoted by b_j the quadrature weights and by $t_k + c_j \Delta t$ the quadrature nodes. It can be proved that for any RK formula (86)-(87), there exists a correspondence between the coefficients b_j , c_j of the formula and the weights and nodes of a Gauss quadrature rule (see, [2, §5.11] for details). For instance, the implicit midpoint method can be seen as a one step implicit RK method based on a 1 point Gauss-Legendre quadrature rule. This corresponds to the Butcher array

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

and can attain order 2. In general, an implicit s-stage Gauss RK method can achieve order $2s$. Similarly, Gauss-Radau and Gauss-Lobatto RK methods can achieve order $2s-1$ and $2s-2$, respectively.

References

[1] E. Hairer, C. Lubich, and G. Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*. Springer, 2006.

[2] J. D. Lambert. *Numerical methods for ordinary differential systems: the initial value problem*. Wiley, 1991.

[3] R. LeVeque. *Finite difference methods for ordinary and partial differential equations*. SIAM, 2007.

[4] A. Quarteroni, R. Sacco, and F. Salieri. *Numerical mathematics*. Springer, 2007.