

Time Discretization — Runge–Kutta and Discontinuous Galerkin

We have seen three examples of time discretization methods: forward Euler, backward Euler, and Crank–Nicolson. All three can be expressed as a “theta-method” for different values of θ . These methods, and many others, are also specific examples of a class of methods called **Runge–Kutta methods**.

1 Runge–Kutta Methods

Runge–Kutta methods are a family of time integration methods, i.e. numerical methods for systems of ODEs. Consider the general model initial value problem

$$\begin{aligned}\frac{d\mathbf{u}}{dt} &= \mathbf{f}(t, \mathbf{u}), \\ \mathbf{u}(0) &= \mathbf{u}_0.\end{aligned}$$

Here, we have generalized to consider potentially nonlinear right-hand side function \mathbf{f} . In our previous example, we had $\mathbf{f}(t, \mathbf{u}) = B\mathbf{u}$, with B a negative-definite matrix.

Runge–Kutta (RK) methods are **one-step** methods: they take as input $\mathbf{u}_i \approx \mathbf{u}(t_i)$, and output an approximate solution $\mathbf{u}_{i+1} \approx \mathbf{u}(t_{i+1})$, where $t_{i+1} = t_i + \Delta t$. Before defining RK methods, we will give a couple of examples. The forward Euler method can be written as

$$\begin{aligned}\mathbf{k}_1 &= \mathbf{f}(t_i, \mathbf{u}_i), \\ \mathbf{u}_{i+1} &= \mathbf{u}_i + \Delta t \mathbf{k}_1.\end{aligned}$$

The backward Euler method can be written as

$$\begin{aligned}\mathbf{k}_1 &= \mathbf{f}(t_i + \Delta t, \mathbf{u}_i + \Delta t \mathbf{k}_1), \\ \mathbf{u}_{i+1} &= \mathbf{u}_i + \Delta t \mathbf{k}_1.\end{aligned}$$

(Note that the equation for \mathbf{k}_1 now becomes **implicit**). We are breaking down these methods using the intermediate (“stage”) variable \mathbf{k}_1 , because in more general Runge–Kutta methods, we can use multiple stages.

For example, the following method is called the explicit midpoint method:

$$\begin{aligned}\mathbf{k}_1 &= \mathbf{f}(t_i, \mathbf{u}_i), \\ \mathbf{k}_2 &= \mathbf{f}(t_i + \tfrac{1}{2}\Delta t, \mathbf{u}_i + \tfrac{1}{2}\Delta t \mathbf{k}_1), \\ \mathbf{u}_{i+1} &= \mathbf{u}_i + \Delta t \mathbf{k}_2.\end{aligned}$$

As the name suggests, this is an *explicit* method, since \mathbf{k}_1 can be computed from \mathbf{u}_i without solving a system, and then \mathbf{k}_2 can be computed once \mathbf{k}_1 is known. This method is second-order accurate.

The number of \mathbf{k}_i variables is called the number of **stages**. We have seen one-stage and two-stage methods, but there are methods of any number of stages. The number of stages is denoted s . The general form of an s -stage Runge–Kutta method is:

$$\begin{aligned}\mathbf{k}_1 &= \mathbf{f}(t_i + c_1 \Delta t, \mathbf{u}_i + \sum_{\ell=1}^s a_{1\ell} \mathbf{k}_\ell), \\ \mathbf{k}_2 &= \mathbf{f}(t_i + c_2 \Delta t, \mathbf{u}_i + \sum_{\ell=1}^s a_{2\ell} \mathbf{k}_\ell), \\ &\vdots \\ \mathbf{k}_s &= \mathbf{f}(t_i + c_s \Delta t, \mathbf{u}_i + \sum_{\ell=1}^s a_{s\ell} \mathbf{k}_\ell), \\ \mathbf{u}_{i+1} &= \mathbf{u}_i + \Delta t (b_1 \mathbf{k}_1 + b_2 \mathbf{k}_2 + \cdots + b_s \mathbf{k}_s).\end{aligned}$$

These methods are determined by the coefficients

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1s} \\ a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1} & a_{s2} & \cdots & a_{ss} \end{pmatrix}, \quad \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_s \end{pmatrix}, \quad \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_s \end{pmatrix}.$$

The matrix $A = (a_{ij})$ is known as the *Runge–Kutta matrix*, the coefficients b_i are called the *weights*, and c_i are called the *abscissas* or *nodes*. These are often arranged in a table called a *Butcher tableau*,

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}
	b_1	b_2	\cdots	b_s

There are conditions on A , \mathbf{b} , and \mathbf{c} to obtain stable and accurate methods; we will not go into this theory.

- If the matrix A is **strictly lower-triangular**, the method is **explicit**. This is because each \mathbf{k}_i depends only on **previous** \mathbf{k}_j . The solution to a system of equations is not required.
- If the matrix A is **lower-triangular**, the method is **diagonally implicit**. Each \mathbf{k}_i depends on previous \mathbf{k}_j and itself. Finding \mathbf{k}_i requires solving a system of size N (the length of the solution vector \mathbf{u}).
- If the matrix A is not lower-triangular, the method is **fully implicit**. Each \mathbf{k}_i may be coupled to all others, and a very big system of size $s \times N$ will need to be solved.

1.1 Examples

The theta method from before can be written as

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \theta & (1-\theta) \\ \hline & \theta & (1-\theta) \end{array}$$

Probably the most popular explicit method is known as “**RK4**”. This is a fourth-order four-stage method with tableau (\cdot represents zero)

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

Writing out this method in full gives

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_i, \mathbf{u}_i), \\ \mathbf{k}_2 &= \mathbf{f}\left(t_i + \frac{1}{2}\Delta t, \mathbf{u}_i + \frac{1}{2}\Delta t \mathbf{k}_1\right), \\ \mathbf{k}_3 &= \mathbf{f}\left(t_i + \frac{1}{2}\Delta t, \mathbf{u}_i + \frac{1}{2}\Delta t \mathbf{k}_2\right), \\ \mathbf{k}_4 &= \mathbf{f}(t_i + \Delta t, \mathbf{u}_i + \Delta t \mathbf{k}_3), \\ \mathbf{u}_{i+1} &= \mathbf{u}_i + \frac{1}{6}\Delta t(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \end{aligned}$$

This method has better accuracy and stability properties than the forward Euler method, but for parabolic problems like the heat equation, implicit methods are still preferred because they can be *unconditionally stable*.

2 Discontinuous Galerkin Methods

We will now discuss another method for temporal discretization based on a finite element approach. We will use the so-called **discontinuous Galerkin** (DG) method. Note that DG is typically used for **spatial discretization** (e.g. instead of the standard finite elements we have seen so far in this course), but it can also be used for temporal discretization.

Recall that the semi-discrete formulation for the heat equation is: find $u_h(t) : [0, T] \rightarrow V_h$ such that

$$(\partial u_h(t)/\partial t, v_h) + a(u_h(t), v_h) = (f, v_h) \quad (1)$$

for all $v_h \in V_h$. The discontinuous Galerkin method uses piecewise polynomials in time to represent u_h . Define the space $\mathcal{P}_q(I)$ on an interval $I = [a, b]$ by

$$\mathcal{P}_q(I) = \left\{ v : [a, b] \rightarrow V_h : v(t) = \sum_{i=0}^q v_i t^i \right\}.$$

Split the time domain $[0, T]$ into a partition of intervals, $[0, T] = \bigcup_{i=1}^K I_i$, $I_i = [t_{i-1}, t_i]$ with $0 = t_0 < t_1 < \dots < t_K = T$. Then, the DG space W_h is

$$W_h = \{v : [0, T] \rightarrow V_h : v|_{I_i} \in \mathcal{P}_q(I_i)\}.$$

Note that we don't place any additional continuity restrictions on $v \in W_h$, so v may be discontinuous at the time sub-interval endpoints t_i . Consider the point t_i , which is the right endpoint of interval I_i and the left endpoint of interval I_{i+1} . Then, we define the notation

$$v^+(t_i) := \lim_{s \rightarrow 0^+} v(t_i + s), \quad v^-(t_i) := \lim_{s \rightarrow 0^-} v(t_i + s).$$

In other words, $v^+(t_i)$ is the trace of v at t_i *from the right*, and $v^-(t_i)$ is the trace of v at t_i *from the left*. We define the **jump** of v at t_i by

$$\llbracket v(t_i) \rrbracket := v^+(t_i) - v^-(t_i).$$

To derive the DG method for (1), choose the test function $v_h \in W_h$ and integrate over the temporal domain $[0, T]$.

$$\int_0^T \left(\frac{\partial u_h}{\partial t}, v_h \right) dt + \int_0^T a(u_h, v_h) dt = \int_0^T (f, v_h) dt.$$

We consider the first term on the left-hand side; for $u_h \in W_h$, this is not actually well defined. Since W_h admits discontinuities, u_h is not differentiable in t at the interval endpoints. To make sense of this term, we break up the integral as

$$\int_0^T \left(\frac{\partial u_h}{\partial t}, v_h \right) dt = \sum_{i=1}^K \int_{t_{i-1}}^{t_i} \left(\frac{\partial u_h}{\partial t}, v_h \right) dt.$$

Focusing on the integral over $[t_{i-1}, t_i]$ and integrating by parts, we obtain

$$\int_{t_{i-1}}^{t_i} \left(\frac{\partial u_h}{\partial t}, v_h \right) dt = - \int_{t_{i-1}}^{t_i} \left(u_h, \frac{\partial v_h}{\partial t} \right) dt + (u_h, v_h)|_{t_i} - (u_h, v_h)|_{t_{i-1}}.$$

Recall that $u_h \in W_h$ is discontinuous at the interval endpoints t_i , and so we should replace the point evaluations at the endpoints $u_h(t_i)$ with a single-valued quantity $\hat{u}_h(t_i)$; this quantity is called the **numerical flux** in DG terminology. In this context, since the information propagates from left to right (with increasing time), it makes sense to $\hat{u}_h(t_i) = u_h^-(t_i)$: we always choose the value from the left interval. This means that $\hat{u}_h(t_0) = u_0$; the numerical flux at the point $t_0 = 0$ is given by the initial condition. Note that the numerical flux is only

used for the *trial function* u_h and not for the test function; the traces for the test function are always taken from within the interval of integration. This gives the formulation

$$- \int_{t_{i-1}}^{t_i} \left(u_h, \frac{\partial v_h}{\partial t} \right) dt + (u_h^-(t_i), v_h^-(t_i)) - (u_h^-(t_{i-1}), v_h^+(t_{i-1})) + \int_{t_{i-1}}^{t_i} a(u_h, v_h) = \int_{t_{i-1}}^{t_i} (f, v_h) dt.$$

This allows for an interval-by-interval solution procedure. Once u_h has been solved for on the interval $[t_{i-2}, t_{i-1}]$, the problem: find $u_h|_{I_i} \in \mathcal{P}_q(I_i)$ such that, for all $v_h \in \mathcal{P}_q(I_i)$,

$$- \int_{t_{i-1}}^{t_i} \left(u_h, \frac{\partial v_h}{\partial t} \right) dt + (u_h^-(t_i), v_h^-(t_i)) + \int_{t_{i-1}}^{t_i} a(u_h, v_h) = \int_{t_{i-1}}^{t_i} (f, v_h) dt + (u_h^-(t_{i-1}), v_h^+(t_{i-1})).$$

Suppose $q = 0$, so $u_h(t)$ is piecewise constant in time. Then, on I_i , $u_h(t) =: u_h^i \in V_h$, and the above formulation simplifies to: find $u_h^i \in V_h$ such that, for all $v_h \in V_h$,

$$(u_h^i, v_h) + \int_{t_{i-1}}^{t_i} a(u_h^i, v_h) dt = \int_{t_{i-1}}^{t_i} (f, v_h) dt + (u_h^{i-1}, v_h)$$

Letting $\Delta t_i = t_i - t_{i-1}$ and rearranging,

$$(u_h^i, v_h) + \Delta t_i a(u_h^i, v_h) = (u_h^{i-1}, v_h) + \int_{t_{i-1}}^{t_i} (f, v_h) dt,$$

which is a simple modification of the backward Euler method (the term $(f(t_i), v_h)$ has been replaced with the average $\int_{t_{i-1}}^{t_i} (f(t), v_h) dt$).

For $q > 0$, the DG-in-time method will result in more complicated systems of equations. These are closely related to fully implicit Runge–Kutta methods.

Now suppose that $q = 1$, so on each time interval, the solution is linear. This means that on each interval $I_i = [t_{i-1}, t_i]$, $u_h(t)$ is determined by its values at the interval endpoints, which we write as $u_0^i, u_1^i \in V_h$. Suppose further that we approximate integrals on each interval $[t_{i-1}, t_i]$ using the simple quadrature rule

$$\int_{t_{i-1}}^{t_i} F(t) dt \approx \frac{\Delta t_i}{2} (F(t_{i-1}) + F(t_i)).$$

Let ϕ_0 be the test function satisfying $\phi_0(t_{i-1}) = v_h \in V_h$ and $\phi_0(t_j) = 0$ for $j \neq i-1$, and let ϕ_1 satisfy $\phi_1(t_i) = v_h \in V_h$ and $\phi_1(t_j) = 0$ for $j \neq i$. Then, using the quadrature approximation,

$$\begin{aligned} - \int_{t_{i-1}}^{t_i} \left(u_h, \frac{\partial \phi_0}{\partial t} \right) dt &\approx \frac{1}{2} ((u_0^i, v_h) + (u_1^i, v_h)) \\ - \int_{t_{i-1}}^{t_i} \left(u_h, \frac{\partial \phi_1}{\partial t} \right) dt &\approx -\frac{1}{2} ((u_0^i, v_h) + (u_1^i, v_h)) \end{aligned}$$

and

$$\begin{aligned} \int_{t_{i-1}}^{t_i} a(u_h, \phi_0) dt &\approx \frac{\Delta t_i}{2} a(u_0^i, v_h) \\ \int_{t_{i-1}}^{t_i} a(u_h, \phi_1) dt &\approx \frac{\Delta t_i}{2} a(u_1^i, v_h). \end{aligned}$$

We obtain a block system, where M is the finite element mass matrix and K is the finite element stiffness matrix,

$$\left[\begin{pmatrix} \frac{1}{2}M & \frac{1}{2}M \\ -\frac{1}{2}M & \frac{1}{2}M \end{pmatrix} + \Delta t \begin{pmatrix} \frac{1}{2}K & 0 \\ 0 & \frac{1}{2}K \end{pmatrix} \right] \begin{pmatrix} u_0^i \\ u_1^i \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\Delta t f(t_{i-1}) + Mu_1^{i-1} \\ \frac{1}{2}\Delta t f(t_i) \end{pmatrix}$$

which can be written in Kronecker form as

$$\left[\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \otimes M + \Delta t \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \otimes K \right] \begin{pmatrix} u_0^i \\ u_1^i \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\Delta t f(t_{i-1}) + Mu_1^{i-1} \\ \frac{1}{2}\Delta t f(t_i) \end{pmatrix}$$

Note that the inverse of the first matrix is given by

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix}^{-1} = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} =: C.$$

Performing a change of variables to k_0, k_1 that satisfy

$$\begin{pmatrix} u_0^i \\ u_1^i \end{pmatrix} = \begin{pmatrix} u_1^{i-1} \\ u_1^{i-1} \end{pmatrix} + \Delta t \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} k_0 \\ k_1 \end{pmatrix},$$

it can be shown that this method is equivalent to the fully implicit Runge–Kutta method

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

which is known as the ‘‘Lobatto IIIC’’ Runge–Kutta method.

3 Automatic time step selection and error control

Using an argument similar to the error estimate for the semi-discrete problem, it is possible to prove that the $q = 0$ fully discrete (DG-in-time) method satisfies the error estimate

$$\|u(t_n) - u_h^n(t_n)\|_{L^2(\Omega)} \leq C' \sqrt{1 + \log \frac{t_n}{\Delta t_n}} \left(\max_{i \leq n} \int_{t_{i-1}}^{t_i} \|\partial u / \partial t\|_{L^2(\Omega)} dt + \max_{t \in [0, t_n]} h^2 \|u(t)\|_{H^2(\Omega)} \right).$$

Here, $T = t_n$ is the final time, and Δt_i is the time step in the i th interval $I_i = [t_{i-1}, t_i]$. In the textbook, the logarithmic term (which grows very slowly) is absorbed into a ‘‘constant’’ C , so from now on, C is used to denote

$$C = C' \sqrt{1 + \log \frac{t_n}{\Delta t_n}}.$$

This is not technically a constant (it depends on the final time and the size of the time steps), but it grows very slowly, and will be dominated by other terms.

The first term in the sum on the right-hand side of the error estimate can be bounded as

$$\int_{t_{i-1}}^{t_i} \|\partial u / \partial t\|_{L^2(\Omega)} dt \leq \Delta t_i \sup_{t \in I_i} \|\partial u / \partial t(t)\|_{L^2(\Omega)}.$$

We use introduce the notation

$$\|\partial u / \partial t\|_{\infty, I_i} := \sup_{t \in I_i} \|\partial u / \partial t(t)\|_{L^2(\Omega)}.$$

The error estimate can then be rewritten as

$$\|u(t_n) - u_h^n(t_n)\|_{L^2(\Omega)} \leq C \left(\max_{i \leq n} \Delta t_i \|\partial u / \partial t\|_{\infty, I_i} + \max_{t \in [0, t_n]} h^2 \|u(t)\|_{H^2(\Omega)} \right).$$

The first term represents the *temporal error* and the second term represents the *spatial error*. If we want the temporal error to be bounded by some prescribed tolerance δ , it makes sense to choose the time steps such that

$$\Delta t_i \|\partial u / \partial t\|_{\infty, I_i} \leq \frac{\delta}{C}.$$

This depends on the quantity $\|\partial u / \partial t\|_{\infty, I_i}$, which we do not know without access to the true solution u . Instead, we can use the first-order estimate

$$\Delta t_i \|\partial u / \partial t\|_{\infty, I_i} \approx \|u_h^i - u_h^{i-1}\|_{L^2(\Omega)},$$

which is computable. So, the time steps can be chosen so that the rule

$$\|u_h^i - u_h^{i-1}\|_{L^2(\Omega)} \approx \frac{\delta}{C}$$

is satisfied. This can be achieved through (automatic) trial and error. A provisional time step $\widetilde{\Delta t}_i$ is chosen (e.g. $\widetilde{\Delta t}_i = \Delta t_{i-1}$). A provisional step is taken to obtain \widetilde{u}_h^i . If \widetilde{u}_h^i satisfies $\|u_h^i - u_h^{i-1}\|_{L^2(\Omega)} \approx \frac{\delta}{C}$, then the time step is accepted. Otherwise, the provisional time step is adjusted (either increased or decreased, and the process is repeated).

A similar procedure can be used to control the spatial error. In this case, we consider a *sequence* of finite element spaces V_h^i , each with mesh size h_i . We want the spatial error to also be bounded by the tolerance δ . For simplicity, assume that $f \equiv 0$. The spatial error on the i th temporal interval is bounded by

$$C \max_i h_i^2 \sup_{t \in I_i} \|u(t)\|_{H^2(\Omega)}.$$

As before, we can't directly compute $\|u(t)\|_{H^2(\Omega)}$, which depends on the exact solution. Instead, note that u satisfies

$$\frac{\partial u}{\partial t} - \Delta u = 0,$$

which, rearranging, gives

$$-\Delta u = f - \frac{\partial u}{\partial t},$$

and by stability of the continuous problem,

$$\|u(t)\|_{H^2(\Omega)} \leq C'' \|\partial u / \partial t(t)\|_{L^2(\Omega)}.$$

So,

$$\sup_{t \in I_i} \|u(t)\|_{H^2(\Omega)} \leq C'' \sup_{t \in I_i} \|\partial u / \partial t(t)\|_{L^2(\Omega)} = C'' \|\partial u / \partial t\|_{\infty, I_i}.$$

We want to choose h_i such that (absorbing the constant C'' into C),

$$h_i^2 \|\partial u / \partial t\|_{\infty, I_i} \leq \frac{\delta}{C}.$$

Approximating $\|\partial u / \partial t\|_{\infty, I_i} \approx \frac{1}{\Delta t_i} \|u_h^i - u_h^{i-1}\|_{L^2(\Omega)}$ as before, we choose h_i such that

$$h_i^2 \|u_h^i - u_h^{i-1}\|_{L^2(\Omega)} \leq \frac{\delta \Delta t_i}{C}.$$