Lecture Note #3 **MTH652: Advanced Numerical Analysis**
Winter 2025 Instructor: Will Pazner

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

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

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

Runge–Kutta (RK) methods are **one-step** methods: they take as input $\boldsymbol{u}_i \approx \boldsymbol{u}(t_i)$, and output an approximate solution $\boldsymbol{u}_{i+1} \approx \boldsymbol{u}(t_{i+1})$, 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

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

The backward Euler method can be written as

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

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

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

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

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

The number of $\boldsymbol{k}_i$ 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:

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

These methods are determined by the coefficients

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

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*,

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

There are conditions on A, $\boldsymbol{b}$, and $\boldsymbol{c}$ 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 $\boldsymbol{k}_i$ depends only on **previous** $\boldsymbol{k}_j$. The solution to a system of equations is not required.
- If the matrix A is **lower-triangular**, the method is **diagonally implicit**. Each $\boldsymbol{k}_i$ depends on previous $\boldsymbol{k}_j$ and itself. Finding $\boldsymbol{k}_i$ requires solving a system of size N (the length of the solution vector $\boldsymbol{u}$).
- If the matrix A is not lower-triangular, the method is **fully implicit**. Each $\boldsymbol{k}_i$ 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

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

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 call *also* be used for temporal discretization.

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

$$(\partial u_h(t)/\partial t, v_h) + a(u_h(t), v_h) = (f, v_h) \tag{1}$$

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

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

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 $0 = t_0 < t_1 < \cdots < t_K = T$. Then, the DG space W_h is

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

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

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

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

$$[\![v(t_i)]\!] := v^+(t_i) - v^-(t_i).$$

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.