RUNGE-KUTTA METHODS

Numerical Methods II, A. DONEV

I will mostly follow LeVeque sec. 5.7.

I already showed examples of 2nd order (trapezoidal, midpoint) and one 4th order (RK4) method.

A general RK method with r stages (so this is a multistage method) to solve the ODE:

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

with numerical approxmation

$$u^k \simeq u(t_k) = u(k\tau)$$

for constant time step size τ

(Later we will talk about adaptive time stepping so τ will become τ_k)

Stage 1:

$$Y_1 = U^n + \bar{\tau} \sum_{j=1}^{r} a_{1j} f(Y_j, t_n + c_j \bar{\tau})$$

This is implicit if any $a_{ij} \neq 0$

So the only explicit option is

$$Y_1 = U^n$$

Stage $k = 1, \ldots, r$

$$Y_k = U^n + \bar{\tau} \sum_{j=1}^{r} a_{kj} f(Y_j, t_n + c_j \bar{\tau})$$

↑ intermediate stage values (e.g., estimate of midpoint)

↖ coefficients ↗

Implicit if $a_{k\{j \geq k\}} \neq 0$

Diagonally implicit if only $a_{kk} \neq 0$

(means we only have to solve nonlinear equation for Y_k)

Finally, the time step update is:

$$U^{n+1} = U^n + \tau \sum_{j=1}^{r} b_j \underbrace{f(Y_j, t_n + c_j \tau)}_{\text{already evaluated above if } a_{kj} \neq 0}$$

So f is evaluated at r points

$$(Y_j, t_n + c_j \tau), \quad j = 1, \ldots, r$$

The coefficients of a particular RK method are represented by a

Butcher tableau

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1r} \\ \vdots & \vdots & \ddots & \vdots \\ c_r & a_{r1} & \cdots & a_{rr} \\ \hline & b_1 & \cdots & b_r \end{array} \equiv \begin{array}{c|c} \vec{c} & \overleftrightarrow{A} \\ \hline & \vec{b}^T \end{array}$$

Explicit if A is lower triangular with zero diagonal

E.g. the RK4 method I showed bases the $\vec{b}$ coefficients on Simpson's rule:

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

If only lower triangle and diagonal of A are non-zero, the method is diagonally implicit — DIRK

Example of 2nd order DIRK:

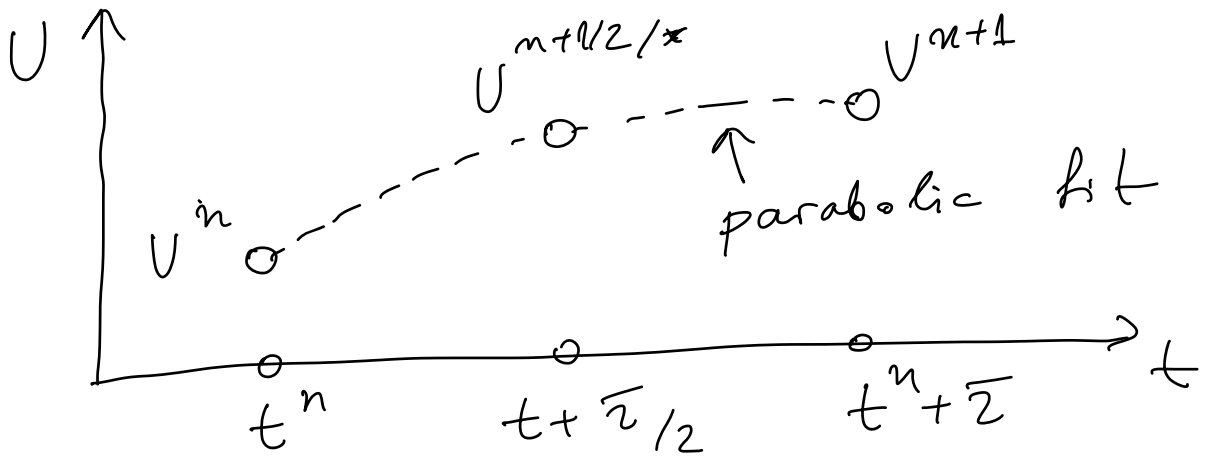
Trapezoidal Rule - Backwards Differentiation formula: TR-BDF2 (see 8.6 in LeVeque)

$$U^{n+1/2,*} = U^n + \frac{\tau}{4}\left(f(U^n) + f(U^{n+1/2,*})\right)$$

This is implicit trapezoidal to midpoint

$$U^{n+1} = \frac{1}{3}\left(4U^{n+1/2,*} - U^n + \tau f(U^{n+1})\right)$$

This update is based on a backwards differentiation formula:



$$u'(t^{n+1}) = f(u(t^{n+1}))$$

Let's fit a parabola through the tree ponts and differentiate that at t^{n+1} to estimate derivative to 2nd order:

$$u'(t^{n+1}) \simeq \frac{U^n + 3U^{n+1} - 4U^{n+1/2,*}}{\bar{\tau}} = f(U^{n+1})$$

which gives the equation for U^{n+1}

This scheme has many nice properties and is often used, as we will explain later

Order conditions

(1) Consistency requires (first-order accuracy)

$$\sum_{j=1}^{r} a_{ij} = c_i, \quad i = 1, \dots, r$$

$$\sum_{j=1}^{r} b_j = 1$$

(2) Second-order requires also:

$$\sum_{j=1}^{r} b_j c_j = 1/2 \quad \text{(non-linear!)}$$

(3) Third order requires also

$$\sum_{j=1}^{r} b_j c_j^2 = 1/3$$

$$\sum_{i,j=1}^{r} a_{ij} b_i c_j = 1/6$$

An example RK3 scheme that plays a special role in practice when solving hyperbolic PDEs (conservation laws) with the finite volume method:

$$f^n = f(U^n, t^n)$$

$$U^* = U^n + \tau f^n \quad \text{(Euler step)}, \quad t^* = t^{n+1}$$

$$U^{**} = \frac{3}{4} U^n + \frac{1}{4} \underbrace{\left(U^* + \tau f^* \right)}_{\text{second Euler step}}$$

$$U^{n+1} = \frac{1}{3} U^n + \frac{2}{3} \underbrace{\left(U^{**} + \tau f^{**} \right)}_{\text{third Euler step}}$$

Observe that each stage is a convex combination of U^n and a forward Euler step. (this is important in the theory)

RK methods have been studied to death (still are...). Here are some important results

① An explicit r-stage RK method with order of accuracy r only exists if $r \leq 4$
(This is why RK4 is special)

② The maximum order of accuracy is $2r$, for a fully implicit RK method (i.e., one has to solve at least r linear systems for all Y_k, $k = 1, \ldots, r$)

Error control & adaptive time stepping

Sophisticated ODE schemes adjust the time step size to control the truncation error. The goal is to meet a certain specified relative or absolute error tolerance. This is not trivial to do, see HW3

E.g., absolute tolerance:

$$\| U^{N \leftarrow \text{final value}} - u(T) \| \leq \mathcal{E}$$

Here I will use $\Delta t \equiv \tau$ (more common notation)

Clearly this will be true if we spread the error uniformly over the time interval $[0, T]$

$$e^k \approx \| U^{k+1} - u((k+1)\tau) \|$$

$\sim$ if we had $U^k = u(k\tau)$

estimated from LTE

(A)

$$\|e^k\| \leq \varepsilon \frac{\Delta t_k}{T} \Rightarrow$$

$$\sum_{k=1}^{N-1} e^k \leq \sum_{k=1}^{N-1} \|e^k\| \leq \varepsilon \frac{\sum_{k=1}^{N-1} \Delta t_k}{T}$$

very pessimistic (safe) global error estimate

but $\sum \Delta t_k = T$

$$e_{global} \leq \varepsilon$$

as desired. This is not necessarily an optimal way to distribute the error but hard to do better.

We want to maximize Δt_k, i.e.,

$$\|e^k\| \simeq \varepsilon \frac{\Delta t_k}{T}$$

How do we find a Δt_k that achieves this desired (max) error?

(B)

Option 1: Richardson extrapolation

(very important technique for ODEs / PDEs)

Idea: Take first a step of duration Δt (or multiple steps), but then also run with half the time step size $\Delta t/2$.

Assume the method is of order p.

$U_{\Delta t}$ = solution with Δt

$U_{\Delta t/2}$ = solution with $\Delta t/2$

u = true solution, assuming we started with the correct solution

$$\begin{cases} u = u_{\Delta t} + C\,\Delta t^{p+1} + O(\Delta t^{p+2}) \\ u = u_{\Delta t/2} + C\left(\frac{\Delta t}{2}\right)^{p+1} + O(\Delta t^{p+2}) \end{cases}$$

same constant $C \sim u^{(p+1)} \simeq u^{(p)} + O(\Delta t^{p+1})$

$$\Rightarrow \quad u_{\Delta t} - u_{\Delta t/2} \simeq C \cdot \Delta t^{p+1} \cdot \left(1 - \frac{1}{2^{p+1}}\right)$$

$$C \Delta t^{p+1} \simeq \frac{2^{p+1}}{2^{p+1}-1} \left(u_{\Delta t} - u_{\Delta t/2}\right)$$

$$\Downarrow$$

$$U \simeq U_{\Delta t} + e_k + O(\Delta t^{p+2})$$

↑ Richardson extrapolation of solution

$$e_k \simeq \frac{2^{p+1}}{2^{p+1}-1} \left(U_{\Delta t} - U_{\Delta t/2}\right)$$

So we increased the order by one via Richardson extrapolation.

This "trick" works for any method, for ODEs or PDEs, in space or in time, etc. Remember it!

Define $\epsilon_0^k = \frac{\Delta t_k}{T} \varepsilon$ ← allowed error for this time step

(D)

This tells us that if we first run (once) Richardson extrapolation with step Δt_k and estimate the LTE e^k, and we then change to step $\tilde{\Delta t}_k$:

$$\|\tilde{e}^k\| \simeq \left(\frac{\tilde{\Delta t}_k}{\Delta t_k}\right)^{p+1} \|e^k\|$$

We want

$$\|\tilde{e}^k\| \simeq \varepsilon \frac{\tilde{\Delta t}_k}{T} = \left(\frac{\tilde{\Delta t}_k}{\Delta t_k}\right) e_0^k$$

$$\Rightarrow \quad \tilde{\Delta t}_k = \Delta t_k \cdot \left(\frac{e_0^k}{\|e^k\|}\right)^{1/p}$$

But if the error was already small enough, $\|e_k\| < e_0^k$, that means we can increase the timestep so that the error is actually e_0^k:

(E)

$$\|\tilde{e}^k\| \simeq \left(\frac{\tilde{\Delta t}_k}{\Delta t_k}\right)^{p+1} \|e^k\| \simeq e_0^k$$

$$\tilde{\Delta t}_k = \left(\frac{e_0^k}{\|e_k\|}\right)^{1/(p+1)} \Delta t_k > \Delta t_k$$

$$\Rightarrow \|\tilde{e}^k\| \simeq e_0^k = \varepsilon \frac{\Delta t_k}{T} < \varepsilon \frac{\tilde{\Delta t}_k}{T}$$

since $\Delta t_k < \tilde{\Delta t}_k$

This leads to this adaptation strategy:

(1) Set the target (absolute) error to

$$e_0^k = \frac{\Delta t_k}{T} \varepsilon$$

(2) Use two different methods (both of order p or one p the other $p+1$) with known and proportional error estimates to estimate LTE e_k.

(F)

(3) Compute an improved estimate for U^{k+1} of order $p+1$ and return this as answer if $\|e_k\| < e_k^0$, and set

$$\Delta t_{k+1} = \Delta t_k \cdot \min\left\{ [5-10], [0.8-0.9] \left(\frac{e_0^k}{\|e^k\|} \right)^{\frac{1}{p+1}} \right\}$$

safety factors

(this time step was OK, so accept it, and increase time step size for next step)

(4) Otherwise (if $\|e_k\| \geq e_0^k$), don't take the step and reduce the time step size

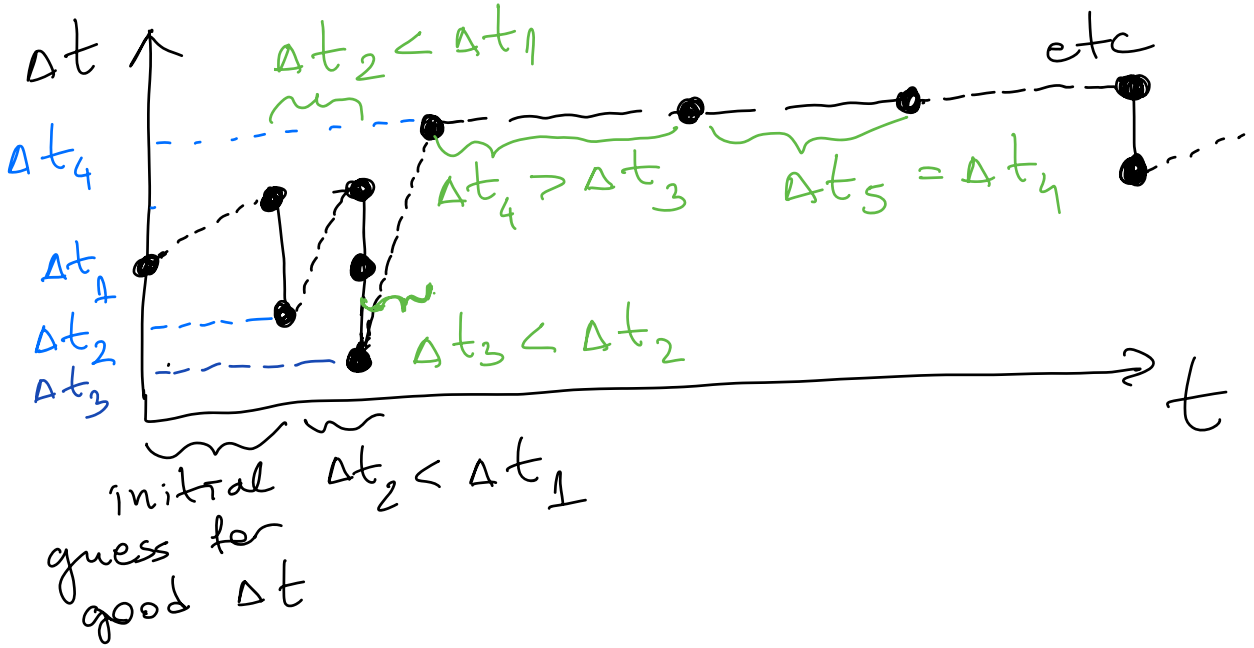
$$\Delta t_k \leftarrow [0.8-0.9] \left(\frac{e_0^k}{\|e_k\|} \right)^{1/p} \cdot \Delta t_k$$

and go back to step 1.

This is a very cautious method. It does not allow us to increase the error beyond what we estimate it to be with current Δt_k.

Important, the estmated error is for the solution of order p, even though we return a solution of order p+1. So this is quite pessimistic!

For homework, to debug, plot Δt



Embedded RK methods

Richardson extrapolation is expensive: each time step we need to take $3r$ stages instead of r to achieve error control.

In practice, we instead construct a set of $r+1$ (or $> r+1$ for very high order) stages that then let us compute both a solution of order p, and a solution of order $p+1$!

$$U_p^{n+1} = u(t_{n+1}) + O(\Delta t^{p+1})$$

$$U = U_{p+1}^{n+1} = u(t_{n+1}) + O(\Delta t^{p+2})$$

↖ best estimate we have (I)

$$\| U_p^{n+1} - u(t_{n+1}) \| \simeq \| U_p^{n+1} - U_{p+1}^{n+1} \| + O(\Delta t^{p+2})$$

$$\| U_{p+1}^{n+1} - u(t_{n+1}) \| = \| e^k \| = O(\Delta t^{p+2})$$

$$\ll \| U_p^{n+1} - U_{p+1}^{n+1} \| = O(\Delta t^{p+1})$$

$$\Rightarrow \| e^k \| \leq \| U_p^{n+1} - U_{p+1}^{n+1} \| = O(\Delta t^{p+1})$$

↑ (pessimistic) error estimate

The method is formally of order p, though it may give a much smaller and controlled error in practice than the original method of order p

E.g. an RK2 embedded in an RK3 method

Bogacki-Shampine RK23

(matlab solver ode23)

$$U^{n+1/2,*} = U^n + \frac{\Delta t}{2} f(U^n, t^n)$$

$$U^{n+3/4,*} = U^n + \frac{3\Delta t}{4} f\left(U^{n+1/2,*}, t^n + \frac{\Delta t}{2}\right)$$

$$U^{n+1} = U_{p+1}^{n+1} = U^n + \Delta t\left(\frac{2}{9} f^n + \frac{1}{3} f^{n+1/2,*} + \frac{4}{9} f^{n+3/4,*}\right)$$

↑ third order estimate, only 3 function evaluations per step

2nd order estimate:

$$U_p^{n+1} = U^n + \Delta t\left(\frac{7}{24} f^n + \frac{1}{4} f^{n+1/2,*} + \frac{1}{3} f^{n+3/4,*}\right) + \frac{1}{8}\Delta t f^{n+1}$$

← reuse next step

Aside: Fully implicit RK methods

As mentioned before, it is possible to get order of accuracy $2r$ with r stages if we use a fully implicit RK method.

For $r=2$, we can get such a 4th order method called the Gauss method, with tableau:

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

Gauss RK2

Consider ODE

$$x'(t) = f(x(t), t) \qquad (A)$$

This method arises by applying Gaussian quadrature with <u>two</u> points:

$$x^{n+1} = x^n + \int_0^{\Delta t} f(x(t), t)\, dt$$

$$\simeq x^n + \frac{\Delta t}{2} \sum_{i=1}^{r} w_i f\left(x\left(\frac{\Delta t}{2}\xi_i + \frac{\Delta t}{2}, \frac{\Delta t}{2}\xi_i + \frac{\Delta t}{2} \right) \right)$$

where for $r = 2$

$$w_{1/2} = 1, \quad \xi_{1/2} = \pm \frac{1}{\sqrt{3}}$$

$$\frac{x^{n+1} - x^n}{\Delta t} = \frac{1}{2}\left[f\left(x\left(\underbrace{\frac{\Delta t}{2} - \frac{\Delta t}{2\sqrt{3}}}, \frac{\Delta t}{2} - \frac{\Delta t}{2\sqrt{3}} \right) \right.\right.$$

$$\left.\left. + f\left(x\left(\underbrace{\frac{\Delta t}{2} + \frac{\Delta t}{2\sqrt{3}}}, \frac{\Delta t}{2} + \frac{\Delta t}{2\sqrt{3}} \right) \right) \right] \quad (B)$$

Denoting $\sqrt{3}\Delta t/6$ = $\frac{\Delta t}{2\sqrt{3}}$

$$F_{1/2} = f\left(X\left(\underbrace{\frac{\Delta t}{2} - \frac{\Delta t}{2\sqrt{3}}}_{t_{1/2}}, \underbrace{\frac{\Delta t}{2} + \frac{\Delta t}{2\sqrt{3}}}_{t_{1/2}}\right)\right)$$

we have

$$X^{n+1} = X^n + \frac{\Delta t}{2}(F_1 + F_2)$$

which is the final step of the Gauss RK2 method.

To turn this into a solvable linear system, we need two more equations, i.e., the two stages of Gauss RK2.

These are

$$\begin{cases} X_1 \simeq X(t_1) = X^n + \Delta t\left(\frac{1}{4}F_1 + \left(\frac{1}{4} - \frac{\sqrt{3}}{6}\right)F_2\right) \\ X_2 \simeq X(t_2) = X^n + \Delta t\left(\left(\frac{1}{4} + \frac{\sqrt{3}}{6}\right)F_1 + \frac{1}{4}F_2\right) \end{cases}$$

Where do these come from? ©

We only need to know F_1 and F_2, i.e., X_1 and X_2, up to $O(\Delta t^3)$ since they are multiplied by Δt. So we need:

$$X_1 = X^n + \int_0^{(1/2-\sqrt{3}/6)\Delta t} f(x(t),t)\,dt + O(\Delta t^3)$$

use linear fit through $(x_1, F_1), (x_2, F_2)$

we only need a first order approximation of this.

This calculation is done in Maple worksheet Euler RK2, and gives

$$X_1 = X^n + \frac{\Delta t}{4} F_1 + \left(\frac{1}{4} - \frac{\sqrt{3}}{6}\right)\Delta t \cdot F_2$$

as it appears in the first stage of Gauss RK2. (D)