

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 approximation

$$u^k \approx u(t_k) \stackrel{\uparrow}{=} u(k\bar{\tau})$$

for constant time step size $\bar{\tau}$

(Later we will talk about adaptive time stepping so $\bar{\tau}$ will become τ_k)

①

Stage 1:

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

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

So the only **explicit** option is

$$Y_1 = U^n$$

Stage $k = 1, \dots, r$

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

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

↑
coefficients

Implicit if $a_{kj} \{ j > k \} \neq 0$

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:

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

already evaluated
 above if $a_{kj} \neq 0$

So f is evaluated at r points

$$(Y_j, t_n + c_j \bar{\tau}), j = 1, \dots, r$$

The coefficients of a particular RK method are represented by a

Butcher tableau

$$\begin{array}{c|ccccc}
 c_1 & a_{11} & - & - & - & a_{1r} \\
 \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\
 c_r & a_{r1} & \cdots & \cdots & a_{rr} & \\
 \hline
 b_1 & - & \cdots & \cdots & \cdots & b_r
 \end{array} = \vec{c} \begin{array}{c|c}
 \xrightarrow{A} & \\
 \hline
 \xrightarrow{B}
 \end{array}$$

Explicit if A is lower triangular with zero diagonal

③

E.g. the RK4 method I showed bases the coefficients on Simpson's rule :

$$\begin{array}{c|ccc} 0 & 0 & \dots & 0 \\ 1/2 & 1/2 & 0 & \\ 1/2 & 0 & 1/2 & 0 \\ 1 & 0 & 0 & 1 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$$

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)

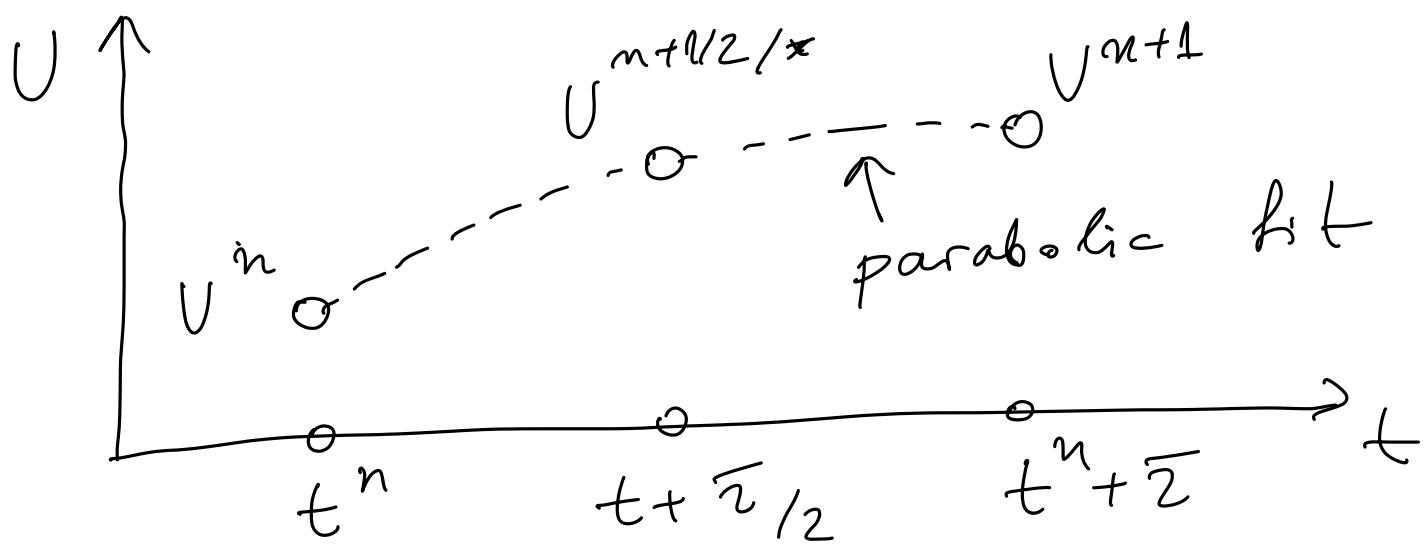
$$U^{n+1/2,*} = U^n + \frac{1}{2} (f(U^n) + f(U^{n+1/2,*}))$$

This is implicit trapezoidal to midpoint

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

④

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 free points and differentiate that at t^{n+1} to estimate derivative to 2nd order:

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

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: $f^n = f(u^n, t^n)$

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

$$u^{**} = \frac{3}{4} U^n + \frac{1}{4} \underbrace{(U^* + \bar{\epsilon} f^*)}_{\text{second Euler step}}$$

$$u^{n+1} = \frac{1}{3} U^n + \frac{2}{3} \underbrace{(U^{**} + \bar{\epsilon} f^{**})}_{\text{third Euler step}}$$

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 , $k = 1, \dots, r$)

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:

$$\| U^{n \leftarrow \text{final value}} - u(T) \| \leq \epsilon$$

Here I will use $\Delta t = \bar{\tau}$

(more common notation)

Clearly this will be true if we spread the error uniformly

over the time interval $[0, T]$
 $e^k \approx \| U^{k+1} - u((k+1)\bar{\tau}) \|$
estimated from LTE

A

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

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

very pessimistic (safe)
global error estimate

$$\text{but } \sum \Delta t_h = T$$

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

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.,

$$\|e^k\| \approx \varepsilon \frac{\Delta t_h}{T}$$

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 $C \sim u^{(P+1)}$
 $\approx u^{(P)} + O(\Delta t^{P+4})$

(C)

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

$$\Rightarrow C \approx \frac{u_{\Delta t} - u_{\Delta t/2}}{\Delta t^{P-1}} \Rightarrow$$

$$U \approx U_{\Delta t} + e_k + O(\Delta t^{P+2})$$

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

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 $c_0^k = \frac{\Delta t_k}{T} e \leftarrow$ 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 $\tilde{\Delta t}_k$:

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

We want

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

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

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 .

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

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

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

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 .

③ Compute an improved estimate for U^{k+1} of order $p+1$ and return this as answer if $\|e_k\| < \epsilon_0^k$, and set

$$\Delta t_{k+1} = \Delta t_k \cdot \min \left\{ \begin{array}{l} [5-10], \\ [0.8-0.9] \left(\frac{\epsilon_0^k}{\|e_k\|} \right)^{\frac{1}{p+1}} \end{array} \right\}$$

safety factors

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

④ Otherwise (if $\|e_k\| \geq \epsilon_0^k$), don't take the step and reduce the time step size

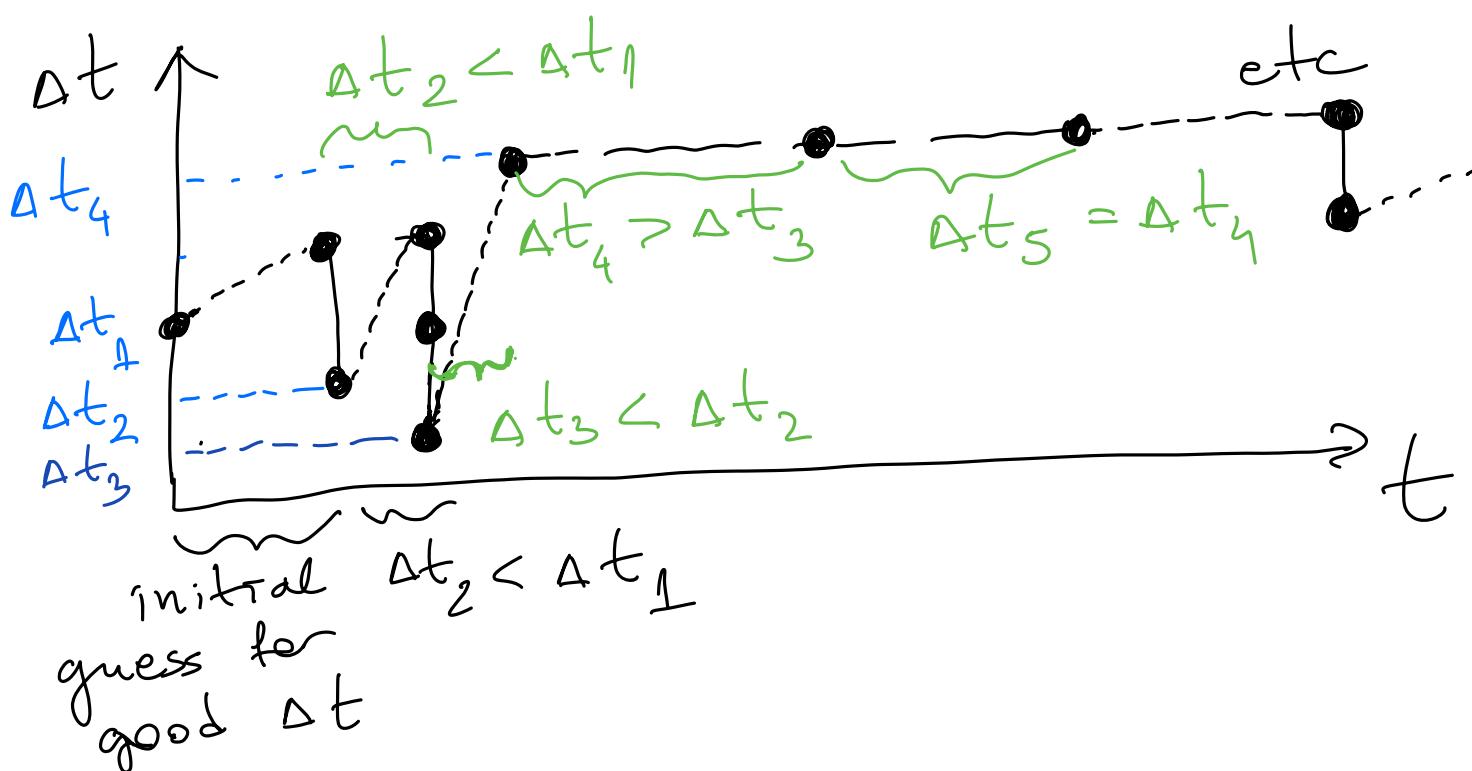
$$\Delta t_k \leftarrow [0.8-0.9] \left(\frac{\epsilon_0^k}{\|e_k\|} \right)^{1/p} \cdot \Delta t_k$$

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 estimated 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



H

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)

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

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

<<

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

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

↑
(pessimistic) error estimate

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



E.g. an RK2 can be used in
an RK3 method

Bogacki-Shampine RK23

(matlab solver ode23)

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

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

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

third order estimate, only

3 function evaluations per step

2nd order estimate:

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