RUNGE-KUTTA METHODS

A. DONEV, COURANT

Following Le Veque section 5.7

We already covered examples of 2^{nd} and one 4^{th} order RK method.

A general RK method with <u>r-stages</u> for the ODE

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

with $u^k \simeq u(k\tau)$

timestep

has the form:

$$Y_1 = U^n + \tau \sum_{j=1}^{r} a_{1j} f(Y_j, t_n + c_j \tau) \quad (2)$$

$\dots$

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

Stage value

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

The function f is evaluated at r points $(Y_j, t_n + c_j \tau)$, $j = 1, \dots, r$

This is represented by a

Butcher tableau:

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

E.g., the RK4 scheme based on Simpson's rule has the tableau:

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array} \leftarrow A \text{ is } \underline{\text{lower triangular}}$$

A strictly lower triangular for explicit methods, and if diagonal is non-zero but lower triangular it is diagonally-implicit RK (**DIRK**) (4)

$a_{ij} = 0$ if $j \geq i$ = explicit

$a_{ij} = 0$ if $j > i$ = DIRK

For DIRK we only need to solve a linear system for Y_k at stage k, instead of solving a big linear system that couples *all* stages: much cheaper!

Example of DIRK:

TR-BDF2 method

$$
\begin{array}{c|ccc}
0 & 0 & 0 & 0 \\
1/2 & 1/4 & 1/4 & 0 \\
1 & 1/3 & 1/3 & 1/3 \\
\hline
 & 1/3 & 1/3 & 1/3
\end{array}
$$

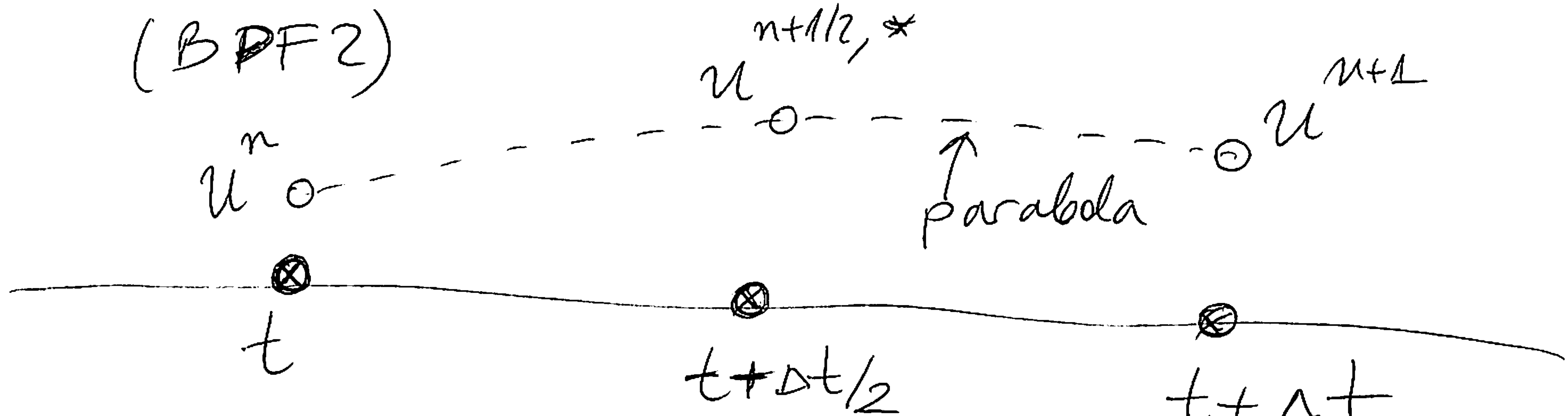
The scheme can be simplified to (see (8.6) in Le Veque)

$$
\begin{aligned}
&\text{trapezoid to midpoint}\;\left\{\; U^{n+1/2,*} = u^n + \frac{\tau}{4}\left(f(U^n) + f(U^{n+1/2,*})\right)\right. \\
&u^{n+1} = \frac{1}{3}\left(4u^{n+1/2,*} - u^n + \tau f(U^{n+1})\right)
\end{aligned}
$$

The predictor step here is a trapezoidal rule (implicit!) to the midpoint. Then the

The second stage is actually a hidden Backward Differentiation Formula

(BDF2)



$$u'(t+\Delta t) = f(u(t+\Delta t))$$

Let's approximate the slope $u'(t+\Delta t)$ by fitting a parabola through the three points and differentiating this at $t+\Delta t$.

$\boxed{\text{exact for parabola} \equiv \text{second order}}$!

Result

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

which gives the second stage

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

Order conditions for RK methods

Consistency requires: (first order)

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

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

Second-order requires further

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

Third order further requires

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

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

An example RK3 scheme that plays a special role for hyperbolic PDEs is: the explicit scheme:

9

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

$$u^{**} = \frac{3}{4} U^{n} + \frac{1}{4} \Big[\underbrace{u^{*} + \tau f^{*}}_{\text{second Euler step}} \Big]$$

$$f^{*} = f(u^{*})$$

$$u^{n+1} = \frac{1}{3} U^{n} + \frac{2}{3} \Big[\underbrace{u^{**} + \tau f^{**}}_{\text{third Euler}} \Big]$$

Observe each stage is a convex linear combination of Euler stages

RK methods have been studied to death. Here are some theorems:

1. An r-stage of order of accuracy r exists only for $r \leq 4$.

2. A fully implicit RK method of r stages can be constructed to have order $2r$.

For fifth order you need six stages.

Going high order is NOT always the best idea as the returns can be diminishing.

Error control & adaptive time steps

Sophisticated ODE schemes adjust the time step size Δt automatically to meet a certain specified error tolerance, either absolute or relative.

This is hard to do (see HW 3).

Assume we want to ensure that

$$\| u^{N = T/\Delta t} - u(T) \| \leq \varepsilon$$

absolute tolerance

One way to guarantee this is to bound the maximum error per unit time, i.e., to spread the error equally over the time interval $[0, T]$. 12

This means we want

e^k = Local Truncation Error LTE(k)

$$\boxed{\|e^k\| \le \varepsilon \frac{\Delta t_k}{T}}$$ local error control

How to choose τ_k to achieve this as a near equality?

Option 1: Richardson extrapolation

Assume we ran the same method with step size Δt and then with $\Delta t/2$ – this could be for just one step or over the whole interval T.

Assume we know the method is of order p.

$$\begin{cases} \text{Step } \Delta t \Rightarrow \text{Solution is } u_{\Delta t} \\ \text{Step } \Delta t/2 \Rightarrow \text{Solution is } u_{\Delta t/2} \end{cases}$$

True solution is u, assuming you started with the exact solution.

14

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

same $\quad \vec{C} \sim u^{(p+1)}$

$$\Rightarrow \quad \vec{u}_{\Delta t} - \vec{u}_{\Delta t/2} \approx \vec{C} \cdot \Delta t^{p+1}\left(1 - \frac{1}{2^p}\right)$$

$$\Rightarrow \quad \vec{C} \approx \frac{\vec{u}_{\Delta t} - \vec{u}_{\Delta t/2}}{2^p - 1}$$

$$\boxed{\vec{u} \approx \vec{u}_{\Delta t} + \left(\frac{\vec{u}_{\Delta t} - \vec{u}_{\Delta t/2}}{2^p - 1}\right)\Delta t^{p+1} + O(\Delta t^{p+2})}$$

more accurate estimator

This is a <u>generic</u> device to get a (<u>p+1</u>)-order scheme from a (p)-order scheme - <u>Richardson extrapolation</u>

In addition to providing a <u>better</u> estimate, this provides for us an <u>upper bound</u> (really an <u>over</u> estmate) of the LTE

$$e^k \approx \vec{C}\,\Delta t^{p+1} = \frac{(\vec{u}_{\Delta t} - \vec{u}_{\Delta t/2})}{2^p - 1}\,\Delta t^{p+1}$$

Imagine we change (adapt) the step size from Δt_k to $\widetilde{\Delta t}_k$ (16)

$$\|\tilde{e}^k\| = \left(\frac{\widetilde{\Delta t}_k}{\Delta t_k}\right)^{p+1} \|e^k\| \leq \varepsilon \frac{\widetilde{\Delta t}_k}{T}$$

A common prescription is to do

$$\begin{cases} \widetilde{\widetilde{\Delta t}}_k = \underbrace{[0.8 - 0.9]}_{\text{safety factor}} \Delta t_k \begin{cases} \left(\dfrac{e_0^k}{\|e^k\|}\right)^{\frac{1}{p+1}} & \text{if } \|e_k\| < e_0^k \\ \left(\dfrac{e_0^k}{\|e^k\|}\right)^{\frac{1}{p}} & \text{otherwise} \end{cases} \\ \widetilde{\Delta t}_k = \min \left\{ \underbrace{[5-10]}_{\text{safety factor}} \Delta t_k , \widetilde{\widetilde{\Delta t}}_k \right\} \end{cases}$$

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

is the <u>target</u> error.

So if the target error is met and we need to increase the step, we use exponent $1/(p+1)$: <u>confirm</u> on your own that this leads to

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

and if the target error is not met, use power $1/p$, which leads to

$$\|\tilde{e}^k\| \approx \frac{\varepsilon \Delta t_k}{T}$$ so we meet the target.

This is now a general adaptation strategy:

1. Set the current target absolute error $e_0^k \sim \Delta t_k$ (e.g. $\frac{\Delta t_k}{T}\varepsilon$)

2. Use two different methods or the same method with Δt and $\Delta t/2$ to estimate the error e_k.

3. Correct the solution using the error estmate (i.e., return the best solution you have) if $\|e_k\| < e_0^k$, and set

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

and continue

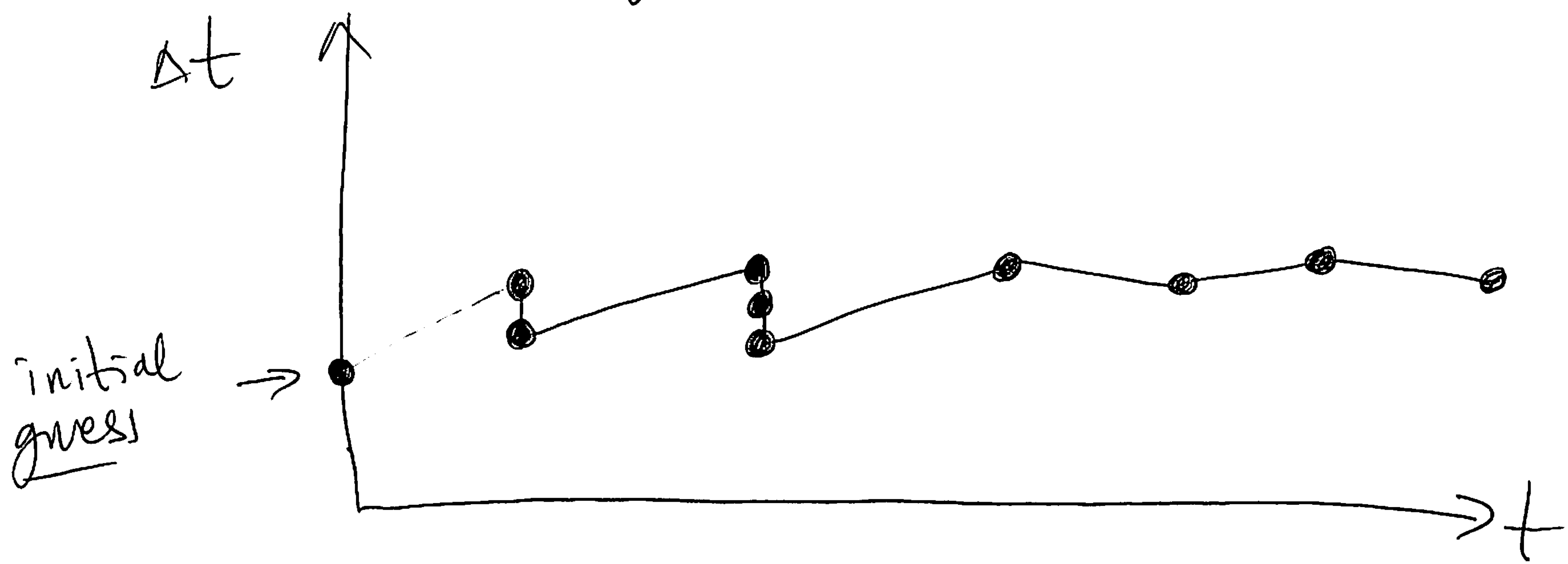
4. Otherwise, if $\|e_k\| > \varepsilon_0^k$, reduce the step size

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

and repeat from step <u>1</u>

Observe that this is a very cautious method. It never allows us to increase the error beyond what it is now, <u>and</u> importantly the <u>error is estimated</u> for the <u>lower-order solution</u>, not the more accurate higher-order solution we actually compute!

To see if this is working correctly it is useful to plot the time step size as a function of time, plotting __each__ repeated step as a point so you can see how many steps are repeated (hopefully not many ≡ if safety factors are OK and problem is not too hard for your method!)



Richardson extrapolation is expensive: each step we need to do $3r$ stages for an r-stage RK method. 21

Instead in practice we use

Embedded RK methods

where the same stages are used to compute both a solution

$u^{n+1,\,low}$ of order p, $LTE = O(\Delta t^{p+1})$

and a more accurate solution

u^{n+1} of order $p+1$, $LTE = O(\Delta t^{p+2})$

The difference is an indication of the error

$$\| u^{n+1,low} - \underset{\substack{\uparrow \\ \text{truth}}}{u} \| \approx \| u^{n+1,low} - u^{n+1} \| \qquad (22)$$

and $\| u^{n+1} - u \| = \| e^k \| \ll \| u^{n+1,low} - u^{n+1} \|$

So we can safely (wastefully?) estimate

$$\| e^k \| \simeq \| u^{n+1,low} - u^{n+1} \| = O(\Delta t^{p+1})$$

and use this to adjust the step size. Note that even though the actual solution/error is $O(\Delta t^{p+1})$ for controlling error we pretend the method is of order p:

$$\boxed{\text{cannot estimate the error of the error estimate!}}$$

Example from HW3 is the

<u>Bogački - Shampine</u> RK 2/3 pair (23)

MATLAB routine ode23

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

$$u^{n+3/4,*} = u^n + \frac{3\Delta t}{4} \underbrace{\boxed{f\left(u^{n+1/2,*}, t^n + \frac{\Delta t}{2}\right)}}_{\equiv f^{n+1/2,*}}$$

3rd order

$$u^{n+1} = u^n + \left(\frac{2}{9} f^n + \frac{1}{3} f^{n+1/2,*} + \underbrace{\boxed{\frac{4}{9} f\left(u^{n+3/4,*}, t^n + \frac{3\Delta t}{4}\right)}}_{\equiv f^{n+3/4,*}}\right)\Delta t$$

2nd order

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

REUSE NEXT STEP!