B) Embedded methods

Idea: Construct a second RK method, which should be:

a) for free (cheap)

b) of different accuracy than the first.

Consider a RK method (b_i, a_{ij}) of order p and with internal stages k_i, $i = 1, 2, \dots, s$.

Build

$$y_{n+1} = y_n + h \sum_{i=1}^{s} b_i k_i$$

$$\hat{y}_{n+1} = y_n + h \sum_{i=1}^{s} \hat{b}_i k_i \quad \text{(same } k_i\text{)}$$

Suppose that $\hat{y}_{n+1}$ has order $\hat{p} < p$ (typically $\hat{p} = p - 1$).

Then

$$|y_{n+1} - \hat{y}_{n+1}| \overset{\Delta \text{ inequality}}{\leq} |y_{n+1} - \varphi_{h_n}(y_n)| + |\hat{y}_{n+1} - \varphi_{h_n}(y_n)|$$

$$= \mathcal{O}(h^{p+1}) + \mathcal{O}(h^{\hat{p}+1})$$

$$= \mathcal{O}(h^{\hat{p}+1})$$

Idea: Use

$$\hat{E}_{n+1}(h_n) = |y_{n+1} - \hat{y}_{n+1}|$$

for the lower order method:

Ex: $y_{n+1} = y_n + \frac{h}{2}\left(f(y_n) + f(y_{n+1})\right)$ (implicit trapez.)

$= y_n + \frac{h}{2}k_1 + \frac{h}{2}k_2$

$k_1 = f(y_n)$, $k_2 = f\left(y_n + \frac{1}{2}h(k_1+k_2)\right)$

Consider $\hat{y}_{n+1}$ / Explicit Euler
\ Implicit Euler

0	0	0	
1	1/2	1/2	Butcher
	1/2	1/2	$\leftarrow y_1$
	1	0	$\leftarrow \hat{y}_1$ (Explicit Euler)

Drawback:

The least accurate method is employed for propagation so that the estimator is indeed an estimator.

In code, one switches the roles of y_{n+1} and $\hat{y}_{n+1}$, i.e. using y_{n+1} as propagation and $\hat{y}_{n+1}$ for error control.
(to avoid waste of computational resources, but only partially theoretically justifiable).

Ex: RK4(5) is an explicit order 5 method with embedded formula of order 4.

MATLAB: ode45
Python: default for scipy.integrate.solve_ivp

2.) Select the time step

Goal: Automatic selection of the steps.

Assume that

$$E_{n+1}(h_n) = C_n h_n^{p+1} .$$

How do we choose h_n s.t. $E_{n+1}(h_n) = Tol$?

$$\Rightarrow h_n = \left(\frac{Tol}{C_n}\right)^{\frac{1}{p+1}} \quad (*) \qquad C_n ?$$

Idea 1: Suppose that $C_n = C_{n-1}$

the error constant, which depends on the differentials of f, is constant in time.

We have

$$E_n(h_{n-1}) = C_{n-1} h_{n-1}^{p+1} ,$$

where we know E_n, h_{n-1}.

$$\Rightarrow C_{n-1} = \frac{E_n(h_{n-1})}{h_{n-1}^{p+1}} = C_n$$

↑ idea 1

By $(*)$ $\Rightarrow h_n = \left(\frac{Tol}{E_n}\right)^{\frac{1}{p+1}} h_{n-1} .$

Algorithm: Once at y_n:

Compute E_n

If $E_n \leq Tol$:

$$h_n = \underbrace{\left(\frac{Tol}{E_n}\right)^{\frac{1}{p+1}}}_{\geq 1} \cdot h_{n-1} \geq h_{n-1}$$

$n \in n+1$

Else:

reject the step, recompute y_n with step

$$h_{n-1} \leftarrow \underbrace{\left(\frac{Tol}{E_n}\right)^{1/p+1}}_{<1} \cdot h_{n-1} < h_{n-1}$$

Drawbacks:

1) Accept $\Rightarrow$ increase of the time step.

(problem with steep variations $\Rightarrow$ a lot of rejected steps $\Rightarrow$ waste of computational resources!)

2) $C_n = C_{n-1}$ is an irrealistic hypothesis to start with. (especially for stiff problems)

Idea 2: Consider that we are in a "tread" of increasing or decreasing difficulty in the integration.

A sensible hypothesis is:

$\frac{C_n}{C_{n-1}}$ is constant

$$\frac{C_n}{C_{n-1}} = \frac{C_{n-1}}{C_{n-2}} \Rightarrow C_n = \frac{C_{n-1}^2}{C_{n-2}}$$

we have

known $\begin{cases} \epsilon_{n-1} = C_{n-2} h_{n-2}^{p+1} \rightarrow C_{n-2} = \frac{\epsilon_{n-1}}{h_{n-2}^{p+1}} \\ \epsilon_n = C_{n-1} h_{n-1}^{p+1} \rightarrow C_{n-1} = \frac{\epsilon_n}{h_{n-1}^{p+1}} \end{cases}$

unknown $\epsilon_{n+1} = C_n h_n^{p+1} = Tol.$

$$\Rightarrow h_n = \left(\frac{Tol}{C_n}\right)^{\frac{1}{p+1}} = \left(\frac{C_{n-2}}{C_{n-1}}\right)^{\frac{1}{p+1}} \cdot \left(\frac{Tol}{C_{n-1}}\right)^{\frac{1}{p+1}}$$

$$= \underbrace{\left(\frac{\epsilon_{n-1}}{\epsilon_n}\right)^{\frac{1}{p+1}} \frac{h_{n-1}}{h_{n-2}}}_{\text{Correction}} \cdot \underbrace{\left(\frac{Tol}{\epsilon_n}\right)^{1/p+1} \cdot h_{n-1}}_{\text{Idea 1}}$$

Rem.: (i) $E_n \leq Tol \not\Rightarrow h_n \geq h_{n-1}$

(ii) One can use this from $n=2$, before Idea 1

(iii) Better for stiff problems.

Final tweaks:

1) safety procedure:

$$h_n = fac \cdot \hat{h}_n ,$$

where $\hat{h}_n$ given by Idea 1/2, $fac = 0,9$.

2) Avoid too rapid increase / decrease of time step.

set:

$$h_n = \min\left(\max\left(\hat{h}_n, \underbrace{facmin}_{\frac{1}{10}} \cdot h_{n-1}\right), \underbrace{facmax}_{5} \cdot h_{n-1}\right)$$

$$\Rightarrow h_n \in [facmin \cdot h_{n-1}, facmax \cdot h_{n-1}]$$

3) There are ways (see Hairer, Nørsett, Wanner) for choosing the first step cleverly.
Otherwise, just use trial and error until you accept.

6. Runge-Kutta-Chebyshev methods

Drawbacks of implicit methods:

1) If d is large ($f : \mathbb{R}^d \to \mathbb{R}^d$)

- think of PDE discretisation - then solving

$$\phi'(z^{(k)})\,\delta z^{(k)} = -\phi(z^{(k)}) \qquad \text{(Newton)}$$

is expensive: needs an iterative solver (GMRES) and a good preconditioner.

2) If h is large, we may not have a good initial guess for Newton, and the norm of ϕ' may be large too, preventing convergence.

3) Implicit methods are usually harder to implement and to debug.

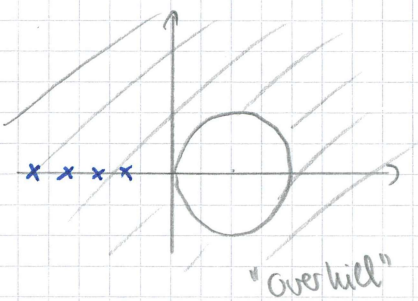
Ex: Heat equation: $\partial_t u - \partial^2_{xx} u = 0$ (PDE)

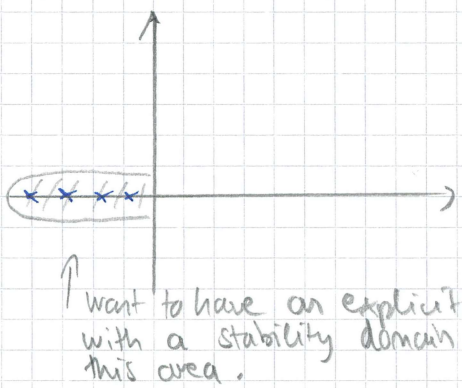
$$u(0,x) = u_0(x)$$
$$u(t,0) = u(t,1) = 0$$

Discretise in space (FEM, FD, Method of lines)

$\Rightarrow$ (ODE) $y'(t) = Ay(t)$ with $A = \frac{1}{(\Delta x)^2}\begin{pmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{pmatrix}$

This problem has large negative real eigenvalues.





Eigenvalues of A are all in $]-\frac{4}{(\Delta x)^2}, 0[$.

Explicit Euler: $h \frac{4}{(\Delta x)^2} \leq 2 \iff h \leq \frac{(\Delta x)^2}{2}$ CFL condition

Rem:
Implicit vs explicit solvers:

implicit: no time step restriction, but large linear systems (linear algebra) to solve

explicit: no linear algebra, but time step restriction.

Idea: We know that for a first order explicit method on s stages it holds:

$$R_s(z) = 1 + z + \underbrace{\sum_{k=2}^{s} \alpha_k z^k}_{\text{extend stability}}$$

$1+z$: accuracy (order)

Ex: Explicit Euler: $s=1$: $R_1(z) = 1+z$ stability function

$$S_1 = \{ z \in \mathbb{C}^-, |R_1(z)| \leq 1 \}$$
$$= \{ z \in \mathbb{C}^-, |1+z| \leq 1 \}$$

$\Rightarrow$ covers the segment $[-2, 0]$ of the negative real axis.

Can we find $\alpha_2, \alpha_3, \dots, \alpha_s$ s.t. $|R_s(z)| \leq 1$ for $z \in [-l_s, 0]$ and l_s large?

Problem 1: Given s, find $\alpha_2, \alpha_3, \dots, \alpha_s$ s.t. $|R_s(z)| \leq 1$ for $z \in [-l_s, 0]$ with l_s maximal.

Lemma: Let $R_s(z) = 1 + z + \sum_{k=2}^{s} d_k z^k$

for $z_s < z_{s-1} < \dots < z_1 < 0$ satisfy

(i) $|R_s(z_i)| = 1$, $i = 1, 2, \dots, s$

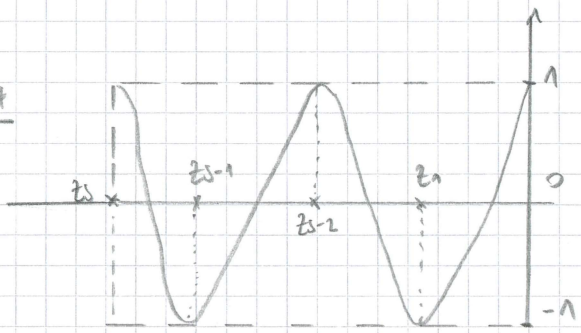
(ii) $R_s(z_{i+1}) = -R_s(z_i)$, $i = 1, 2, \dots, s-1$

(iii) $|R_s(z)| \leq 1$ $\forall z \in [z_s, 0]$.

Then $R_s(z)$ is the unique solution of Problem 1 with $l_s = |z_s|$.

Ex:

$s = 4$



Proof: Let $\bar{R}_s(z) = 1 + z + \sum_{k=2}^{s} \bar{d}_k z^k$ satisfy $|\bar{R}_s(z)| \leq 1$

for $z \in [-\bar{l}_s, 0]$ with $\bar{l}_s > l_s$.

Set $d_s(z) = R_s(z) - \bar{R}_s(z) = z^2 \sum_{k=2}^{s} (d_k - \bar{d}_k) z^{k-2}$

$R_s(z)$ oscillates s times between -1 and 1 in the interval $[-l_s, 0[$.
($R_s(z_i) = (-1)^i$)

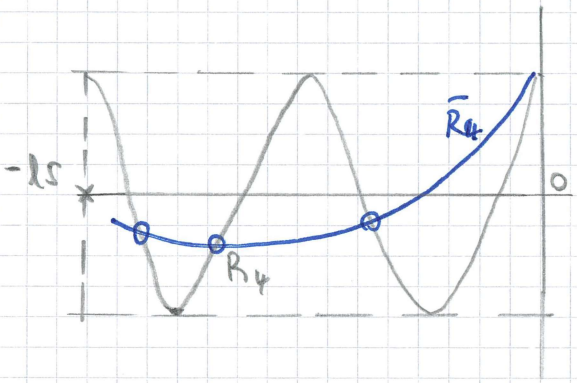
and $|\bar{R}_s(z)| \leq 1$ in $[-\bar{l}_s, 0]$ with $\bar{l}_s > l_s$

Hence, $\bar{R}_s$ and R_s cross in at least $s-1$ points in the interval $[-l_s, 0[$.

$\Rightarrow$ $d_s(z)$ has at least $s-1$ zeros in $[-l_s, 0[$.

But $d_S(z)$ has (by definition) at most $S-2$ zeros away from $z=0$.

$\Rightarrow$ Contradiction $\lightning$ #



$S=4$

- R_4 and $\bar{R}_4$ cross

$\Rightarrow$ 3 zeros for $d_4(z)$.

The solution of Problem 1, i.e. a polynomial satisfying (i)-(iii), is given by the Chebyshev polynomials.

Defn: (Chebyshev polynomial)

$$T_S(x) = \cos(S \cdot \arccos(x)) \quad \text{for } x \in [-1,1].$$

Alternatively, by a recurrence relation:

$$T_0(x) = 1, \; T_1(x) = x,$$

$$T_n(x) = 2x\,T_{n-1}(x) - T_{n-2}(x).$$

Theorem: $R_S(z) = T_S\left(1 + \frac{z}{S^2}\right)$ solves Problem 1 with $l_S = 2S^2$.

Proof: Exercises.