More advanced time-stepping methods

Explicit Euler, $y_{n+1} = y_n + hf(t_n, y_n)$

only evaluates f at one point.

Why not use more?

To do this properly, we must first discuss

Numerical integration:

- approximating $\int_a^b f(x)\,dx$ note: no y

For this, we also need some

Interpolation

- "opposite of discretization"

Formally:

Formally:

Given $\{f_j\}_0^N$ on grid $\{x_j\}_0^N$, find a continuous function f with the interpolating property $f(x_j) = f_j$

Find f in "nice" class of functions, like polynomials or trigonometric functions (Fourier analysis)

Suppose

$$P_n(x) = c_0 + c_1 x + \cdots + c_n x^n$$

$$\begin{cases} n+1 \text{ coefficients } c_j \\ n+1 \text{ interpolation conditions } P_n(x_j) = f_j \end{cases}$$

Can write this as

$$\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & & & & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}}_{=:A} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}}_{=:c} = \underbrace{\begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{bmatrix}}_{=:\underline{f}}$$

i.e. $A c = \bar{f}$ with Vandermonde matrix A.

Works fine for $n \lessapprox 5$ if $x_i \neq x_j$ but is problematic for large n:

A becomes ill-conditioned (see course in numerical linear algebra)

Better: use other basis functions than $\{1, x, x^2, \dots, x^n\}$.

Lagrange interpolation

Still want $f_j = P(x_j) = c_0 + c_1 x_j + \dots + c_n x_j^n$

but write it instead as

$$P(x) = \sum_{i=0}^{n} \varphi_i(x) f_i$$

with basis functions $\{\varphi_0, \varphi_1, \dots, \varphi_n\}$

Smart choice: Lagrange basis:

$$\begin{cases} \varphi_i \text{ polynomial of degree } n, \text{ and} \\ \varphi_i(x_j) = \delta_{ij} = \begin{cases} 1, & i=j \\ 0, & i \neq j \end{cases} \end{cases}$$

Kronecker delta

Then

$$\varphi_i(x_j) f_i = \begin{cases} f_i , & i=j \\ 0 , & i \neq j \end{cases}$$

so

$$\sum_{i=0}^{n} \varphi_i(x_j) f_i = \varphi_j(x_j) f_j = f_j$$

$\Rightarrow \quad P(x_j) = f_j$ $\quad \Rightarrow$ interpolation condition satisfied by construction!

φ_i is easy to write down:

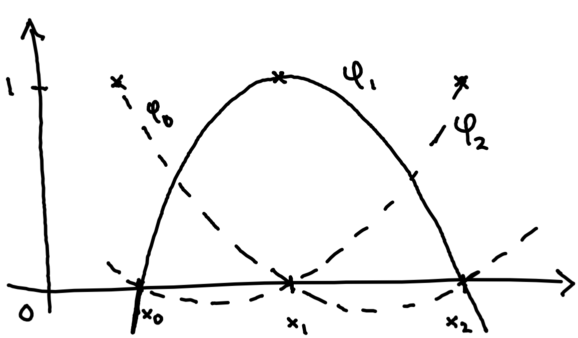
<u>Ex</u> $n=2$ and $i=1$

Then grid $\{x_0, x_1, x_2\}$ and e.g. $\begin{cases} \varphi_1(x_0) = \varphi_1(x_2) = 0 \\ \varphi_1(x_1) = 1 \end{cases}$

Satisfied by

$$\varphi_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \begin{array}{l} \leftarrow \text{gives 2nd-deg. polynomial and correct zeroes} \\ \leftarrow \text{scaling factor} \end{array}$$

Appearance:



General form, $n=2$:

$$P_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f_1 + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f_2$$

Can be evaluated very efficiently by clever code!

Numerical integration

Cannot compute $\int_a^b f(x)\,dx$ in general, but

can compute $\int_a^b P(x)\,dx$ if P polynomial

Idea: Interpolate $f(x) \approx P(x)$ and approx. $\int f(x)dx \approx \int P(x)dx$

We get

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n} \underbrace{\int_a^b \varphi_i(x)\,dx}_{=:w_i} \cdot f(x_i)$$

The weights w_i do not depend on f!

$\Rightarrow$ compute once and for all

General numerical integration method:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i)$$

where w_i depends on grid $\{x_i\}_0^n$ and choice of basis.

Runge-Kutta methods

Back to IVPs: $\dot{y}(t) = f(t, y(t))$

Integrate:

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} \dot{y}(t)dt = \int_{t_n}^{t_{n+1}} f(t, y(t))dt$$

Not helpful since y unknown in

But we can approximate!

Use $a = t_n$, $b = t_{n+1}$

since n is taken

and grid $\{t_n + c_i h\}_{i=1}^{s}$

not 0 due to tradition

Then

$$\int_{t_n}^{t_{n+1}} f(t,y(t))\,dt \approx \sum_{j=1}^{s} \underbrace{hb_j}_{\text{"}w_j\text{"}} f(t_n+c_jh,\, y(t_n+c_jh))$$

Now approximate $Y_j \approx y(t_n+c_jh)$, $j=1,\dots,s$.

This gives the explicit Runge-Kutta method

$$y_{n+1} = y_n + \sum_{j=1}^{s} b_j h f(t_n+c_jh,\, Y_j)$$

We update Y_j using the same idea:

Abbreviate $hY_j' = hf(t_n+c_jh,\, Y_j)$

and use $$Y_i = y_n + \sum_{j=1}^{i-1} a_{i,j} h Y_j'$$

with the coefficients $a_{i,j}$ to be determined.

We call $\{Y_j\}$ <u>stage values</u>

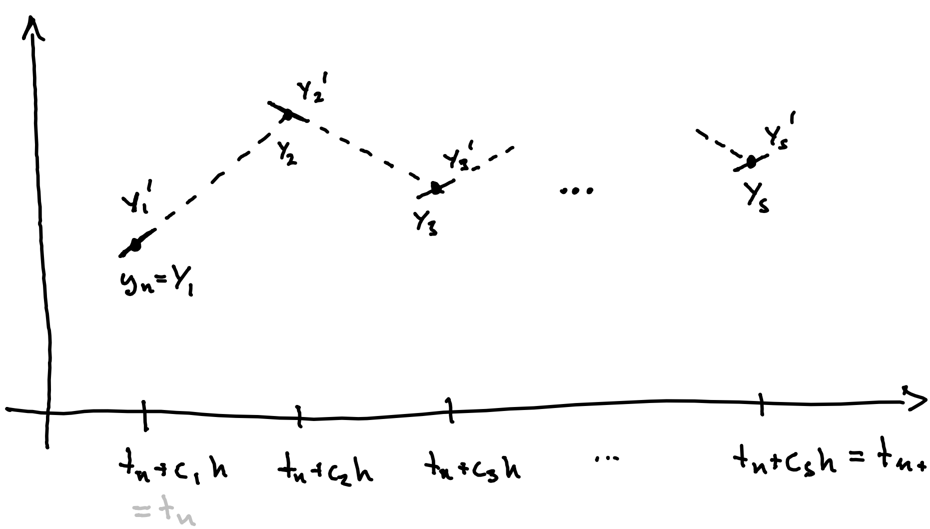
and $\{Y_j'\}$ <u>stage derivatives</u>

But note that the $'$ does not denote differentiation. It just signifies an approximation of a derivative.

Full method:

$$\begin{cases} Y_i = y_n + \sum_{j=1}^{i-1} a_{i,j} h Y_j' , & i=1,\dots,s, \\ hY_j' = hf(t_n + c_j h, Y_j) , & j=1,\dots,s, \\ y_{n+1} = y_n + \sum_{j=1}^{s} b_j h Y_j' \end{cases}$$

Intuitive idea:



Then we use all the information $\{Y_j'\}$ to better approximate $y(t_{n+1})$.

Compact representation of <u>nodes</u> $\{c_i\}$, <u>weights</u> $\{b_j\}$ and <u>coefficients</u> $\{a_{i,j}\}$ in

Butcher tableau

$$
0 = \begin{array}{c|cccc}
c_1 & 0 & 0 & \cdots & 0 \\
c_2 & a_{2,1} & 0 & \cdots & 0 \\
\vdots & \vdots & & \ddots & \\
c_s & a_{s,1} & a_{s,2} & \cdots & 0 \\
\hline
 & b_1 & b_2 & \cdots & b_s
\end{array}
\qquad \text{or} \qquad
\begin{array}{c|c}
c & A \\
\hline
 & b^T
\end{array}
$$

Note that A is lower-triangular so far, for <u>explicit</u> RK-methods.

Finding A, b and c

$\longrightarrow$

Finding A, b and c

Simplifying assumption:

$$c_i = \sum_{j=1}^{s} a_{i,j} \quad \text{(row sums of A)}$$

$\Rightarrow Y_j$ 1st-order approx. to $y(t_n + c_j h)$

Not necessary, but simplifies matters.

Let's analyze the simplest case with $s=2$. (s=1 is explicit Euler)

$$\begin{array}{c|cc} c_1 & 0 & 0 \\ c_2 & a_{2,1} & 0 \\ \hline & b_1 & b_2 \end{array} \quad \text{with} \quad \begin{aligned} c_1 &= 0 \\ c_2 &= a_{2,1} \end{aligned}$$

Formulas:

$$\begin{cases} hY_1' = hf(t_n, y_n) \\ hY_2' = hf(t_n + c_2 h, y_n + a_{2,1} hY_1') \\ y_{n+1} = y_n + b_1 hY_1' + b_2 hY_2' \end{cases}$$

Let's expand in Taylor series around (t_n, y_n):

$$hY_2' = hf(t_n, y_n) + hf_t(t_n, y_n) \cdot c_2 h$$
$$+ hf_y(t_n, y_n) \cdot a_{2,1} \underbrace{hY_1'}_{hf(t_n, y_n)} + O(h^3)$$

$$\Rightarrow \quad y_{n+1} = y_n + h(b_1 + b_2) f(t_n, y_n)$$
$$+ h^2 b_2 \left(c_2 f_t + a_{21} f_y f \right) + O(h^3)$$

Now do the same for the exact solution:

$$y'(t) = f(t, y(t)) \quad \Rightarrow \quad y'' = f_t + f_y y' = f_t + f_y f$$

$$\Rightarrow \quad y(t+h) = y(t) + hy'(t) + \frac{h^2}{2} y''(t) + O(h^3)$$
$$= y + hf + \frac{h^2}{2}(f_t + f_y f) + O(h^3)$$

The local error is $\hat{y}_{n+1} - y(t_n + h)$,

where $\hat{y}_{n+1}$ is the result of the method

starting from $(t_n, y(t_n))$. So, $\rightarrow$

So, for order 1, local error $\mathcal{O}(h^2)$,

the h-terms must match, i.e. $\boxed{b_1 + b_2 = 1}$

For order 2, also the h^2-terms must match:

$$b_2 c_2 = \frac{1}{2} \quad \text{and} \quad b_2 a_{2,1} = \frac{1}{2}$$

With $c_2 = a_{2,1}$ it is the same equation.

We get the <u>order conditions</u>

$$\begin{cases} b_1 + b_2 = 1 & \text{order 1} \\ \quad b_2 a_{2,1} = \frac{1}{2} & \text{order 2} \end{cases}$$

2 eq., 3 unknowns $\Rightarrow$ 1-parameter family of sols.:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2b} & \frac{1}{2b} & 0 \\ \hline & 1-b & b \end{array} \qquad (b \neq 0)$$

Ex "Heun's method" with $b = \frac{1}{2}$:

$$\begin{cases} hY_1' = hf(t_n, y_n) \\ hY_2' = hf(t_n + h, y_n + hY_1') \\ y_{n+1} = y_n + \frac{1}{2}(hY_1' + hY_2') \end{cases}$$

Order 2 and explicit

Compare to the 2nd-order implicit trapezoidal rule

$$y_{n+1} = y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_n + h, y_{n+1})\right)$$

(Explicit is cheaper than implicit but not always better. The trap. rule has better stability properties.)

For order 3, we need at least 3 stages $\{Y_1, Y_2, Y_3\}$.

Taylor series + match terms $\Rightarrow$

Order conditions with $c_i = \sum_j a_{i,j}$

$$b_1 + b_2 + b_3 = 1$$ order 1

$$\left.\begin{aligned} b_2 c_2 + b_3 c_3 &= 1/2 \\ b_2 c_2^2 + b_3 c_3^2 &= 1/2 \end{aligned}\right\}$$ order 2

$$b_3 a_{3,2} c_2 = 1/6$$ order 3

Ex

RK3:

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 1 & -1 & 2 & 0 \\ \hline & 1/6 & 4/6 & 1/6 \end{array}$$

$\longrightarrow$

Even higher order

"The" RK method, RK4 (1895):

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

4 stages, order 4

Order s with s stages only possible if $s \leq 4$.

Order 5 needs 6 stages.

High (>3) order conditions

Taylor series gets (incredibly) complicated

instead →

Instead, B-series (1970s), a graph theory approach. (Way too advanced for this course!)

Still, $\overset{b}{s} + \overset{A}{\frac{s(s-1)}{2}} \approx s^2$ parameters

but $\sim 2^p$ order conditions for order p

<u>Ex.</u> $s = 11$, $p = 8$

$\Rightarrow$ 66 parameters, 200 conditions

(Still many methods, many conditions coincide.)

Order 10: 1205 conditions.

Main point: difficult, still active research.

Embedded RK methods

Combine two RK methods with almost the same stages

Ex. RK34:

$$hY_1' = hf(t_n, y_n)$$

$$hY_2' = hf(t_n + \tfrac{h}{2}, y_n + \tfrac{1}{2}hY_1')$$

$$hY_3' = hf(t_n + \tfrac{h}{2}, y_n + \tfrac{1}{2}hY_2')$$

new $\rightarrow$ $$hZ_3' = hf(t_n + h, y_n - hY_1' + 2hY_2')$$

$$hY_4' = hf(t_n + h, y_n + hY_3')$$

$$\begin{array}{c|ccccc}
0 & 0 & & & & \\
1/2 & 1/2 & 0 & & \bigcirc & \\
1/2 & 0 & 1/2 & 0 & & \\
1 & -1 & 2 & 0 & 0 & \\
1 & 0 & 0 & 1 & 0 & 0 \\
\hline
y_{n+1}: & 1/6 & 2/6 & 2/6 & 0 & 1/6 \\
z_{n+1}: & 1/6 & 4/6 & 0 & 1/6 & 0 \\
 & Y_1' & Y_2' & Y_3' & Z_3' & Y_4'
\end{array}$$

$$y_{n+1} = y_n + \frac{1}{6}\left(hY_1' + 2hY_2' + 2hY_3' + hY_4'\right)$$

$$z_{n+1} = y_n + \frac{1}{6}\left(hY_1' + 4hY_2' + hZ_3'\right)$$

y_{n+1} 4th-order approx. and z_{n+1} 3rd-order approx.

Assume $y_n = y(t_n)$ exact:

$$\Rightarrow \|y_{n+1} - y(t_{n+1})\| = \mathcal{O}(h^5), \quad \|z_{n+1} - y(t_{n+1})\| = \mathcal{O}(h^4)$$

$$\Rightarrow \|z_{n+1} - y_{n+1}\| \le \|z_{n+1} - y(t_{n+1})\| + \|y(t_{n+1}) - y_{n+1}\| = \mathcal{O}(h^4) - \mathcal{O}(h^5)$$
$$= \mathcal{O}(h^4)$$

and $\|z_{n+1} - y(t_{n+1})\| \le \|z_{n+1} - y_{n+1}\| + \mathcal{O}(h^5)$

$\Rightarrow \|z_{n+1} - y_{n+1}\|$ is a 3rd-order estimate of the local error $z_{n+1} - y(t_{n+1})$

Important: We don't know the exact error, but this error estimate is easy to compute

In practice: Estimate error of z_{n+1} but use the more accurate y_{n+1} as the actual approx. The y_{n+1}-error will be smaller than the z_{n+1}-error, so the error estimate is over-estimating.

Adaptive time-stepping / local error control

Idea: make sure the error estimate $r_{n+1} := \|z_{n+1} - y_{n+1}\|$ satisfies $r_n \approx TOL$ in every step by changing h

Then also the local error $\approx TOL$

Assume $r_n = C h_n^p$ with fixed C.

Then
$$r_{n+1} = C h_{n+1}^p = \frac{r_n}{h_n^p} h_{n+1}^p$$

To get $r_{n+1} = TOL$ we must choose $h_{n+1} = \left(\frac{TOL}{r_n}\right)^{1/p} h_n$

Simplest adaptive strategy for order p-1 error estimate:

$$h_{n+1} = \left(\frac{TOL}{r_n}\right)^{1/p} h_n$$

We can do this significantly better via control theory techniques — see Project 1.

Implicit Runge-Kutta methods

ERK : lower-triangular A

IRK : not so

General form:
$$\begin{cases} hY_i' = hf\left(t_n + c_i h,\ y_n + \sum_{j=1}^{s} a_{i,j} hY_j'\right) \\ y_{n+1} = y_n + \sum_{i=1}^{s} b_i hY_i' \end{cases}$$

note

Implicit => must solve equation for $\{Y_i'\}$

Can also write
$$\begin{cases} Y_i = y_n + \sum_{j=1}^{s} a_{i,j} hY_j' \\ hY_i' = hf(t_n + c_i h, Y_i) \end{cases}$$

<u>Ex.</u> Implicit Euler : $\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$

Implicit midpoint method : $\begin{array}{c|c} 1/2 & 1/2 \\ \hline & 1 \end{array}$
(order 2)

General 1-stage IRK:

$$hY_1' = hf(t_n + c_1 h,\ y_n + a_{11} hY_1')$$

$$y_{n+1} = y_n + b_1 hY_1'$$

$$= y_n + b_1 hf(\cdots)$$

$$\begin{pmatrix} f \sim f(t_n, y_n) \\ f_t \sim f_t(t_n, y_n) \\ f_y \sim f_y(t_n, y_n) \end{pmatrix}$$

Order conditions via Taylor series:

$$y_{n+1} = y_n + b_1 h \underbrace{\left(f + f_t \cdot c_1 h + f_y \cdot a_{11} hY_1' + \mathcal{O}(h^2)\right)}_{Y_1'}$$

$$= y_n + h \cdot b_1 f + h^2 \cdot \left[b_1 c_1 f_t + b_1 a_{11} f_y \left(f + f_t c_1 h + f_y a_{11} hY_1' + \mathcal{O}(h^2)\right)\right]$$

$$= y_n + hb_1 f + h^2 \left(b_1 c_1 f_t + b_1 a_{11} f_y f\right) + \mathcal{O}(h^3)$$

Exact:

$$y(t_{n+1}) = y(t_n) + hf + \frac{h^2}{2}\left(f_t + f_y f\right) + \mathcal{O}(h^3)$$

Match terms!

$$\rightarrow \begin{cases} b_1 = 1 & \text{order } 1 \\ c_1 = a_{11} = \frac{1}{2} & \text{order } 2 \end{cases}$$

We have shown consistency orders for many methods. We also have

> **Theorem** (without proof)
>
> All Runge-Kutta methods that are consistent of order p are also convergent of order p.

(For multistep methods (later), this does not hold!)

We now consider also stability:

if the exact solution stays bounded, for which h does the numerical approximation also stay bounded?

Stability

Consider linear test equation: $y' = \lambda y$

$$\rightarrow \quad hY_i' = h\lambda\left(y_n + \sum_{j=1}^{s} a_{i,j} hY_j'\right)$$

Stack these s equations with

$$h\overline{Y}' = \begin{bmatrix} hY_1' \\ hY_2' \\ \vdots \\ hY_s' \end{bmatrix} \quad \text{and} \quad \mathbb{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

Then

$$h\overline{Y}' = h\lambda \mathbb{1} y_n + h\lambda A h\overline{Y}'$$

i.e. $$h\overline{Y}' = (I - h\lambda A)^{-1} h\lambda \mathbb{1} y_n$$

$$\Rightarrow y_{n+1} = y_n + \sum_{j=1}^{s} b_j hY_j' = \left(1 + h\lambda b^T (I - h\lambda A)^{-1} \mathbb{1}\right) y_n$$

$$=: R(h\lambda)\, y_n$$

Theorem

Every R-K method applied to $y' = \lambda y$ leads to

$$y_{n+1} = R(h\lambda)\, y_n$$

with the stability function

$$R(z) = 1 + z b^T (I - zA)^{-1} \mathbb{1}.$$

If the s-stage method is explicit, R is a polynomial of degree s, otherwise R is rational.

Note: theorem for further general analysis only. Everyone who tries to use it on the exam mess something up.

Better for us:

Ex. (2021-01 exam Q1, part 1)

$$\begin{array}{c|cc} 1/2 & 1/2 & 0 \\ 1 & a & 0 \\ \hline & 1/2 & 1/2 \end{array} \quad \Leftrightarrow \quad \begin{aligned} hY_1' &= h\lambda\left(y_n + \tfrac{1}{2} hY_1'\right) \\ hY_2' &= h\lambda\left(y_n + a hY_1' + \cancel{0 \cdot hY_2'}\right) \\ y_{n+1} &= y_n + \tfrac{1}{2} hY_1' + \tfrac{1}{2} hY_2' \end{aligned}$$

Solve for hY_1', insert into hY_2' and y_{n+1}:

$$hY_1' = \frac{h\lambda}{1-\frac{1}{2}h\lambda}\, y_n = \frac{2h\lambda}{2-h\lambda}\, y_n$$

simplifying usually helps

$$hY_2' = h\lambda\left(y_n + ahY_1'\right) = \frac{2h\lambda - (h\lambda)^2 + 2a(h\lambda)^2}{2-h\lambda}\, y_n$$

$$= \frac{2h\lambda + (h\lambda)^2(2a-1)}{2-h\lambda}\, y_n$$

$$y_{n+1} = y_n + \frac{h\lambda}{2-h\lambda}\, y_n + \frac{h\lambda + (h\lambda)^2(a-\frac{1}{2})}{2-h\lambda}\, y_n$$

$$= \frac{2 + h\lambda + (h\lambda)^2(a-\frac{1}{2})}{2-h\lambda}\, y_n = R(h\lambda)\, y_n$$

Note: rational R since implicit RK. Polynomial if explicit. (Why?)

<u>A-stability</u>

Recall: <u>stability region</u> $D = \{ z \in \mathbb{C} \,;\, |R(z)| \leq 1 \}$

<u>A-stable</u> if $D \supset \mathbb{C}^-$ (left half-plane)

Theorem

No explicit R-K method is A-stable.

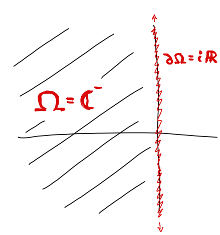
Proof:

The stability function R of an ERK is a polynomial, so $|R(z)| \to \infty$ as $z \to -\infty$. □

Proof technique for IRK: the maximum principle (complex analysis)

If f is analytic on Ω then $|f|$ attains its maximum on $\partial\Omega$ (the boundary of Ω).

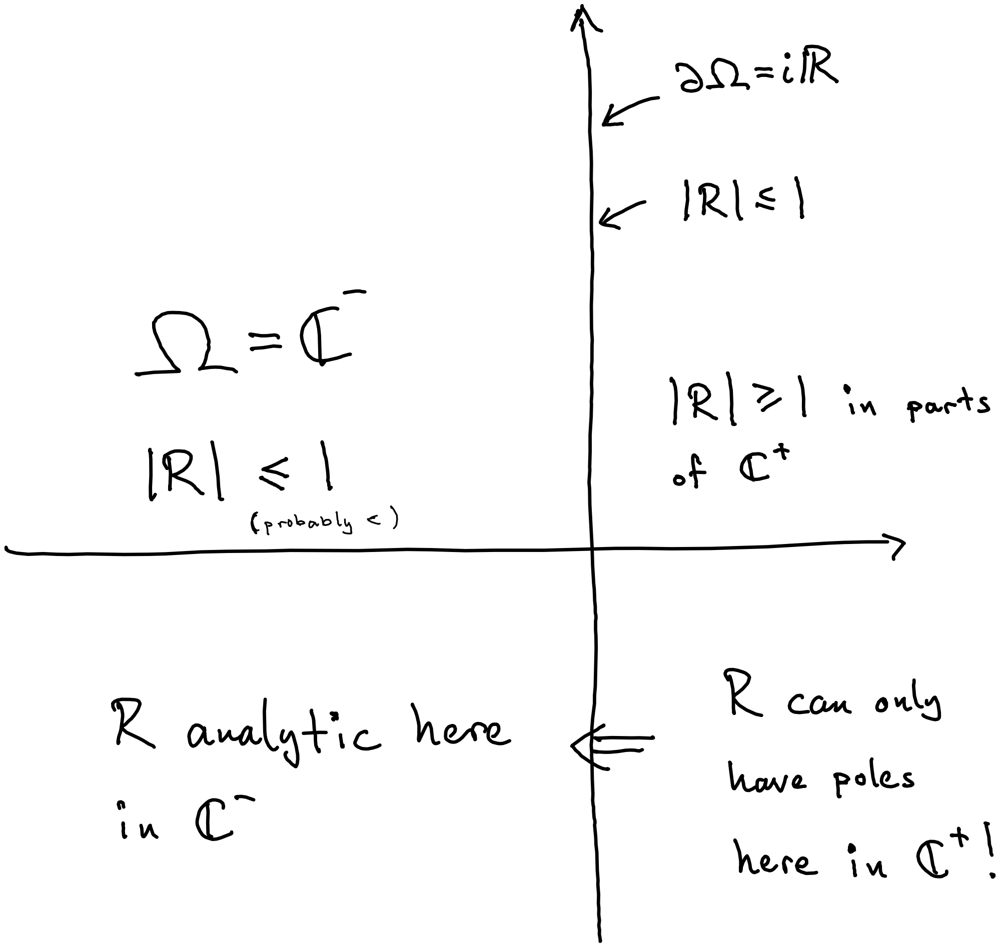
Here, R rational $\Rightarrow$ analytic on Ω if no poles in Ω



Theorem

An IRK with stability function R is A-stable iff

- all poles of R are in $\mathbb{C}^+$ (right half-plane)
- $|R(i\omega)| \leq 1 \quad \forall\, \omega \in \mathbb{R}$ (boundary of $\mathbb{C}^+$).


$\partial\Omega = i\mathbb{R}$
$|R| \leq 1$
$\Omega = \mathbb{C}^-$
$|R| \leq 1$
(probably <)
$|R| \geq 1$ in parts of $\mathbb{C}^+$
R analytic here in $\mathbb{C}^-$
R can only have poles here in $\mathbb{C}^+$!

Ex. (continued)

$$R(z) = \frac{2+z+z^2(a-\frac{1}{2})}{2-z}$$

- single pole at $z=2 \in \mathbb{C}^+$. OK!

- $$R(i\omega) = \frac{2-\omega^2(a-\frac{1}{2})+i\omega}{2-i\omega}$$

(Recall $|z|^2 = (\mathrm{Re}\, z)^2 + (\mathrm{Im}\, z)^2$

$$\Rightarrow |R(i\omega)|^2 = \frac{(2-\omega^2(a-\frac{1}{2}))^2+\omega^2}{2^2+\omega^2}$$

$$= \frac{4-4\omega^2(a-\frac{1}{2})+\omega^4(a-\frac{1}{2})^2+\omega^2}{4+\omega^2}$$

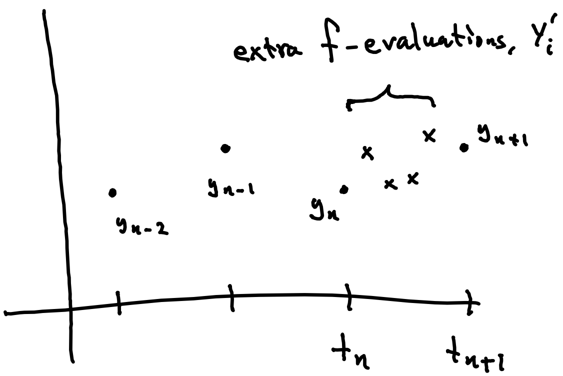
For $|R(i\omega)|^2 \le 1$ we must have $a=\frac{1}{2}$, or the numerator grows faster than the denominator.

With $a=\frac{1}{2}$, $|R(i\omega)|^2 = \frac{4+\omega^2}{4+\omega^2} = 1 \le 1 \quad \forall \omega$.

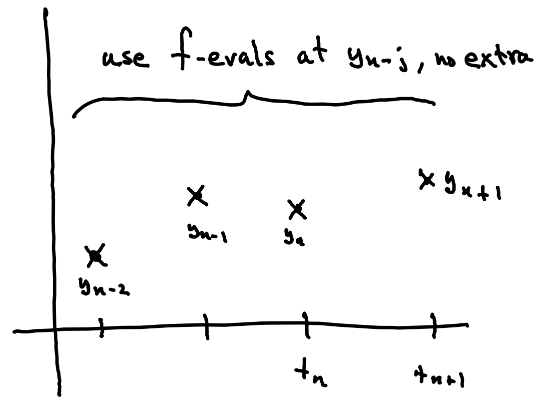
$\therefore$ A-stable method iff $a=\frac{1}{2}$.

Linear multistep methods

Previously:



Now:



General form: $y_{n+1} = \Phi(f, h, y_0, y_1, \dots, y_n, y_{n+1})$

Form of <u>linear</u> (k-step) multistep method (LMM):

$$\boxed{\sum_{j=0}^{k} a_j y_{n+j} = h \sum_{j=0}^{k} b_j f(t_{n+j}, y_{n+j})}$$

Note: know $y_0, \dots, y_{n+k-1}$, look for y_{n+k}

(traditional notation)

Note 1 : not unique

Normalization : $a_k = 1$ or $\sum_{j=0}^{k} b_j = 1$

Note 2 :

Explicit iff $b_k = 0$

Ex. (also one-step methods)

Expl. Euler : $y_{n+1} - y_n = hf(t_n, y_n)$

Impl. Euler : $y_{n+1} - y_n = hf(t_{n+1}, y_{n+1})$

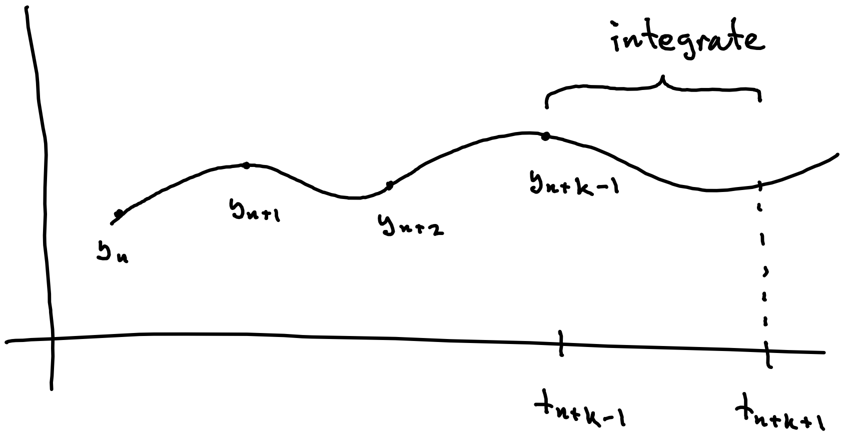
Trap. rule : $y_{n+1} - y_n = \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right)$

Adams methods

Idea :

$$y(t_{n+k}) - y(t_{n+k-1}) = \int_{t_{n+k-1}}^{t_{n+k}} f(\tau, y(\tau))\, d\tau$$

Approx. $f(\tau, y(\tau))$ by interpolating $y_n, y_{n+1}, \dots, y_{n+k-1}$ at $t_n, t_{n+1}, \dots, t_{n+k-1}$



Suppose P interpolating polynomial with $\deg P = k-1$

i.e. $P(t_{n+j}) = f(t_{n+j}, y(t_{n+j}))$, $j = 0, \ldots, k-1$

Then if f "nice" (no proof, see numerical analysis course):

$$P(\tau) = f(\tau, y(\tau)) + \mathcal{O}(h^k)$$

$$\Rightarrow y(t_{n+k}) = y(t_{n+k-1}) + \int_{t_{n+k-1}}^{t_{n+k}} P(\tau)\, d\tau + \mathcal{O}\left(h^{k+1}\right)$$

since $t_{n+k} - t_{n+k-1} = h$.

Leads to $\longrightarrow$

$\boxed{\text{Adams-Bashforth methods, order } k: \quad y_{n+k} = y_{n+k-1} + \sum_{j=0}^{k-1} b_j h f(t_{n+j}, y_{n+j})}$

where $b_j = \frac{1}{h} \int_{t_{n+k-1}}^{t_{n+k}} \varphi_j(\tau)\,d\tau$ given by

integrating the Lagrange basis polynomials φ_j.

Ex. (study Q)

$k=1 \Rightarrow y_{n+1} = y_n + hf(t_n, y_n)$ (Expl. Euler)

$k=2 \Rightarrow y_{n+2} = y_{n+1} + \frac{3}{2} hf(t_{n+1}, y_{n+1}) - \frac{1}{2} f(t_n, y_n)$

If we include $y(t_{n+k})$ in the interpolation we get the implicit

$\boxed{\text{Adams-Moulton methods, order } k+1: \quad y_{n+k} = y_{n+k-1} + \sum_{j=0}^{k} b_j h f(t_{n+j}, y_{n+j})}$ (other coeff. b_j)

[Interlude with some info on J.C. Adams (1819-1892) and the Voyager 2 space probe (1977-)]

Start-up issues

A 2-step method goes from y_n, y_{n+1} to y_{n+2}.

But in the first step, we only have y_0 and not y_1.

Solution: use another method which is accurate enough.

Can do e.g. AB1 → AB2 → AB3 but this does not work for AB4.

Automatic in good software, but still research on how to do this most efficiently.

[Demo which shows errors for AB1 and AB2.]

Order of LMM

The order of consistency is p if the local error

$$\ell(y) = \sum_{j=0}^{k} a_j\, y(t_{n+j}) - h \sum_{j=0}^{k} b_j\, y'(t_{n+j}) = O\left(h^{p+1}\right)$$

To get conditions on a_j, b_j, Taylor-expand at t_n, y_n:

$$\ell(y) = \sum_{j=0}^{k} a_j \left(y(t_n) + jh\, y'(t_n) + \frac{(jh)^2}{2} y''(t_n) + \cdots \right)$$

$$- \sum_{j=0}^{k} h\, b_j \left(y'(t_n) + jh\, y''(t_n) + \frac{(jh)^2}{2} y^{(3)}(t_n) + \cdots \right)$$

$$= y(t_n) \cdot \sum a_j + h\, y'(t_n) \sum \left(j a_j - b_j \right)$$

$$+ h^2 y''(t_n) \sum \left(\frac{j^2}{2} a_j - j b_j \right)$$

$$+ \cdots$$

$\ell(y)$ should be $\mathcal{O}(h^{p+1})$ for <u>all</u> solutions!

Insert polynomials:

$$y(t) = 1 \quad \Rightarrow \quad y' = y'' = \dots = 0 \quad \Rightarrow \quad \sum a_j = 0$$

$$y(t) = t \quad \Rightarrow \quad y' = 1,\ y'' = y^{(3)} = \dots = 0 \quad \Rightarrow \quad \sum j a_j - b_j = 0$$

$$\vdots$$

$$y(t) = t^p \quad \Rightarrow \quad \sum \frac{j^p}{p!} a_j - \frac{j^{p-1}}{(p-1)!} b_j = 0$$

$y(t) = t^{p+1}$ gives us h^{p+1}-terms, nothing more cancels.

$\therefore$ $\ell(y) = \mathcal{O}(h^{p+1})$ iff $\ell(y) = 0$ for all polynomials of degree $\leq p$.

$\Rightarrow$ Easy (exam) test for consistency order:

insert $y(t) = t^m$ and $y(t) = m t^{m-1}$.

Note: enough to test with $t_{n+j} = jh$:

$\ell(t^q)$ is a polynomial of degree q

If $\ell(t^q)(jh) = 0$ for <u>all</u> j then

$\ell(t^q)$ must be the zero polynomial.

Thus $\ell(t^q)(t_n + jh) = 0$ for <u>any</u> t_n.

Theorem (summary)

A k-step LMM has consistency order p iff

$$\sum_{j=0}^{k} j^m a_j = m \sum_{j=0}^{k} j^{n-1} b_j \quad , \; m = 0, 1, \ldots, p$$

The order p is maximal iff

$$\sum_{j=0}^{k} j^{p+1} a_j \neq (p+1) \sum_{j=0}^{k} j^p b_j \; .$$

Alternatively, the order is p if the method is <u>exact</u> for polynomial solutions of degree $\leq p$.

Stability

Trickier for multi-step methods.

- Finite step stability: which h such that approximation of sol. to $y' = \lambda y$ stays bounded?

Same idea as previously for R-K, determines which problems the method is useful for.

For multistep methods, also

- Zero-stability: does the approximation to the sol. of $y' = 0$ stay bounded?

$\boxed{\text{Necessary for convergence.}}$ R-K methods are always zero-stable, LMMs not necessarily so.

Apply LMM to $y' = 0$:

$$\sum_{j=0}^{k} a_j y_{n+j} = 0$$

This is a <u>linear</u> <u>difference</u> <u>equation</u>

(approximating a <u>differential</u> equation)

Recap: General solution to

$$a_k y_k + a_{k-1} y_{k-1} + \cdots + a_1 y_1 + a_0 y_0 = 0$$

can be found via the <u>characteristic equation</u>

$$\underbrace{a_k w^k + a_{k-1} w^{k-1} + \cdots + a_1 w + a_0}_{\text{characteristic polynomial}} = 0$$

$\longrightarrow$

$$a_k w^k + a_{k-1} w^{k-1} + \cdots + a_1 w + a_0 = 0$$

Assume it has the roots $w_1, \ldots, w_k$

- If w_i is a single root, $y_n = w_i^n$ is a solution
- If w_i is a root of multiplicity j, then

$$y_n = w_i^n, \quad y_n = n w_i^n, \quad \ldots, \text{ and } y_n = n^{j-1} w_i^n$$

are all solutions.

The general solution is a linear combination of the solutions corresponding to each root.

So if we want $|y_n| \le C$, we must satisfy the

$$\boxed{\begin{array}{l} \underline{\text{Root condition}} \\ \bullet \text{ all roots must satisfy } |w_i| \le 1 \\ \bullet \text{ if } w_i \text{ is a multiple root then } |w_i| < 1. \end{array}}$$

($|n^k w_i^n| \to \infty$ if $k > 0$ and $|w_i| = 1$, but $\to 0$ if $|w_i| < 1$.)

Adams methods: $y_{n+k} = y_{n+k-1} + \sum_{j=0}^{k} b_j h f_{n+j}$

$\rightarrow \quad y_{n+k} - y_{n+k-1} = 0$

Characteristic eq.: $w^k - w^{k-1} = 0$

$\Leftrightarrow \quad w^{k-1}(w-1) = 0$

0 is a root with multiplicity $k-1$

1 is a root with multiplicity 1

$\Rightarrow$ Root condition satisfied

$\therefore$ All Adams methods are zero-stable

<u>Dahlquist equivalence theorem</u>

A multistep method is convergent of order $p \geq 1$ iff it is zero-stable and consistent of order p.

Dahlquist's first barrier

The maximal order of a zero-stable k-step LMM is

- $k+1$ if k odd
- $k+2$ if k even

$\Big\}$ implicit methods

- k if $b_k/a_k \leq 0$ $\Big\} \supset$ explicit methods

A-stability

Same idea as for R-K:

Apply to $y' = \lambda y$. Does y_n stay bounded?

Char. eq. $\sum_{j=0}^{k} a_j w^j - h\lambda \sum_{j=0}^{k} b_j w^j = 0$

Roots w_i depend on $h\lambda$!

For each $h\lambda$, check if root condition satisfied

A-stable if $h\lambda \in \mathbb{C}^- \Rightarrow |y_n| \leq C$.

Dahlquist's second barrier

The maximal order of an A-stable LMM is 2.

Among the A-stable 2nd-order LMM's, the trapezoidal rule has the smallest error constant.

(No such problem for R-K methods.)

Useful "almost A-stable" LMMs:

Backward differentiation formulae (BDF)

$$\sum_{j=0}^{k} a_j y_{n+j} = h f(t_{n+k}, y_{n+k})$$

Many ways to find $\{a_j\}$ (try Taylor series!)

Zero-stable only for $1 \le k \le 6$

Order k

[MATLAB illustration of BDF stability regions]

Important property: stable for all $h\lambda \in \mathbb{R}^-$

Stiff problems characterized by large negative eigenvalues

$\Rightarrow$ BDF works well for stiff problems.