Numerical Methods for Differential Equations

Chapter 2: Runge–Kutta and Multistep Methods

Tony Stillfjord, Gustaf Söderlind

Numerical Analysis, Lund University

Contents

1. Runge–Kutta methods

Given an IVP $y' = f(t, y),\ y(0) = y_0$ use numerical integration to approximate integrals

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(\tau, y(\tau))\,\mathrm{d}\tau \quad \Rightarrow$$

$$y(t_{n+1}) \approx y(t_n) + h \sum_{j=1}^{s} b_j f(t_n + c_j h, y(t_n + c_j h))$$

Let $\{Y_j\}_{j=1}^{s}$ denote numerical approximations to $\{y(t_n + c_j h)\}_{j=1}^{s}$
A *Runge–Kutta method* then has the form

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

The explicit Runge–Kutta computational process

Sample vector field to obtain *stage derivatives*

$$hY_j' = hf(t_n + c_j h, Y_j)$$

at *stage values*

$$Y_i = y_n + \sum_{j=1}^{i-1} a_{i,j} hY_j'$$

and advance solution one step by a linear combination

$$y_{n+1} = y_n + \sum_{j=1}^{s} b_j hY_j'$$

An s-stage RK method has *nodes* $\{c_i\}_{i=1}^{s}$ and *weights* $\{b_j\}_{i=1}^{s}$

The *Butcher tableau* of an explicit RK method is

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

Simplifying assumption $\quad c_i = \sum_{j=1}^{s} a_{i,j} \quad$ (row sums of RK matrix)

$\Longrightarrow$ every stage value Y_j is 1st-order approx. of solution $y(t_j)$

A two-stage explicit RK method has *three "free" coefficients*
The simplifying assumption determines the nodes

$$
\begin{aligned}
hY_1' &= hf(t_n, y_n) \\
hY_2' &= hf(t_n + c_2 h, y_n + h\, a_{21}\, Y_1') \\
y_{n+1} &= y_n + [\, b_1\, hY_1' + b_2\, hY_2' \,]
\end{aligned}
$$

Butcher tableau

$$
\begin{array}{c|cc}
0 & 0 & 0 \\
c_2 & a_{21} & 0 \\
\hline
 & b_1 & b_2
\end{array}
\qquad c_1 = 0,\ c_2 = a_{21}
$$

Derivation of two-stage ERK's

Using $hY_1' = hf(t_n, y_n)$, expand hY_2' in Taylor series around t_n, y_n

$$\begin{aligned} hY_2' &= hf(t_n + c_2 h, y_n + h\, a_{21} f(t_n, y_n)) \\ &= hf + h^2 \left[c_2 f_t + a_{21} f_y f\right] + \mathcal{O}(h^3) \end{aligned}$$

Insert into $y_{n+1} = y_n + b_1\, hY_1' + b_2\, hY_2'$ and use $c_2 = a_{21}$ to obtain

$$y_{n+1} = y_n + (b_1 + b_2)hf + h^2 b_2 c_2 (f_t + f_y f) + \mathcal{O}(h^3)$$

Expand exact solution in Taylor series and match terms

$$\begin{aligned} y' &= f \\ y'' &= f_t + f_y y' = f_t + f_y f \\ y(t+h) &= y + hf + \frac{h^2}{2}(f_t + f_y f) + \mathcal{O}(h^3) \end{aligned}$$

One-parameter family of 2nd order two-stage ERK's

Match terms to get *conditions for order 2*

$$
\begin{aligned}
b_1 + b_2 &= 1 \qquad \text{(consistency)} \\
b_2 c_2 &= 1/2
\end{aligned}
$$

Note *Consistent RK methods are always convergent*

Two equations, three unknowns $\Rightarrow$ there is a one-parameter family of 2nd order two-stage ERK methods with Butcher tableau

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

Put $b = 1$ to get the Butcher tableau

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

$$\begin{aligned} hY_1' &= hf(t_n, y_n) \\ hY_2' &= hf(t_n + h/2, y_n + hY_1'/2) \end{aligned}$$

$$y_{n+1} = y_n + hY_2'$$

Second order two-stage explicit Runge–Kutta (ERK) method

Put $b = 1/2$ to get

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

$$
\begin{aligned}
hY_1' &= hf(t_n, y_n) \\
hY_2' &= hf(t_n + h, y_n + hY_1')
\end{aligned}
$$

$$y_{n+1} = y_n + (hY_1' + hY_2')/2$$

Second order two-stage ERK, compare to the trapezoidal rule

Third order three-stage ERK

Conditions for 3rd order ($c_2 = a_{21}$; $c_3 = a_{31} + a_{32}$)

$$
\begin{aligned}
b_1 + b_2 + b_3 &= 1 \\
b_2 c_2 + b_3 c_3 &= 1/2 \\
b_2 c_2^2 + b_3 c_3^2 &= 1/3 \\
b_3 a_{32} c_2 &= 1/6
\end{aligned}
$$

Classical RK3

0	0	0	0
1/2	1/2	0	0
1	−1	2	0
	1/6	2/3	1/6

Nyström scheme

0	0	0	0
2/3	2/3	0	0
2/3	0	2/3	0
	1/4	3/8	3/8

Exercise

Construct the Butcher tableau for the 3-stage Heun method.

$$
\begin{aligned}
hY_1' &= hf(t_n, y_n) \\
hY_2' &= hf(t_n + h/3, y_n + hY_1'/3) \\
hY_3' &= hf(t_n + 2h/3, y_n + 2hY_2'/3)
\end{aligned}
$$

$$y_{n+1} = y_n + (hY_1' + 3hY_3')/4$$

Is the method of order 3?

The "original" RK method (1895)

$$
\begin{aligned}
hY_1' &= hf(t_n, y_n) \\
hY_2' &= hf(t_n + h/2, y_n + hY_1'/2) \\
hY_3' &= hf(t_n + h/2, y_n + hY_2'/2) \\
hY_4' &= hf(t_n + h, y_n + hY_3')
\end{aligned}
$$

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

Classical RK4 . . .

Butcher tableau

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

Note s-stage ERK methods of order $p = s$ exist only for $s \leq 4$

There is no 5-stage ERK of order 5

An s-stage ERK method has $s + s(s-1)/2$ coefficients to choose, but there are overwhelmingly many order conditions

\# of available coefficients

stages s	1	2	3	4	5	6	7	8	9	10	11
coefficients	1	3	6	10	15	21	28	36	45	55	66

\# of order conditions and min # of stages to achieve order p

order p	1	2	3	4	5	6	7	8	9	10
conditions	1	2	4	8	17	37	85	200	486	1205
min stages	1	2	3	4	6	7	9	11	?	?

2. Embedded RK methods

Two methods in a single Butcher tableau (RK34)

$$
\begin{aligned}
hY_1' &= hf(t_n, y_n)\\
hY_2' &= hf(t_n + h/2, y_n + hY_1'/2)\\
hY_3' &= hf(t_n + h/2, y_n + hY_2'/2)\\
hZ_3' &= hf(t_n + h, y_n - hY_1' + 2hY_2')\\
hY_4' &= hf(t_n + h, y_n + hY_3')
\end{aligned}
$$

$$
\begin{aligned}
y_{n+1} &= y_n + \frac{1}{6}\left(hY_1' + 2hY_2' + 2hY_3' + hY_4'\right) && \text{order 4}\\
z_{n+1} &= y_n + \frac{1}{6}\left(hY_1' + 4hY_2' + hZ_3'\right) && \text{order 3}
\end{aligned}
$$

The difference $y_{n+1} - z_{n+1}$ *can be used as an error estimate*

Use an embedded pair, e.g. RK34

Local error estimate $r_{n+1} := \|y_{n+1} - z_{n+1}\| = \mathcal{O}(h^4)$

Adjust the step size h to make local error estimate equal to a prescribed *error tolerance* TOL

Simplest step size updating scheme

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

makes $r_n \approx \mathrm{TOL}$

Time step adaptivity using local error control

There are many state-of-the-art embedded ERK methods, e.g.

- Dormand–Prince DOPRI45 (1980)
- Dormand–Prince DOPRI78 (1981)
- Cash–Karp CK5 (1990)

Advanced adaptivity uses discrete control theory and digital filters

$$h_{n+1} = \rho_n \cdot h_n$$
$$\rho_n = \left(\frac{\mathrm{TOL}}{r_{n+1}}\right)^{\beta_1/p} \left(\frac{\mathrm{TOL}}{r_n}\right)^{\beta_2/p} \rho_{n-1}^{-\alpha}$$

PI control, ARMA filters &c., via control parameters $(\beta_1, \beta_2, \alpha)$

3. Implicit Runge–Kutta methods (IRK)

In ERK, the matrix A in the tableau *is strictly lower triangular*

In IRK, A may have *nonzero diagonal elements* or even be full

$$hY_i' = hf\big(t_n + c_i h, y_n + \sum_{j=1}^{s} a_{i,j}\, hY_j'\big)$$

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i\, hY_i'$$

The method is implicit and *requires equation solving* to compute the stage derivatives $\{Y_i'\}_{i=1}^{s}$

Implicit Runge–Kutta methods...

In stage value – stage derivative form

$$Y_i = y_n + \sum_{j=1}^{s} a_{i,j}\, hY'_j$$

$$hY'_i = hf(t_n + c_i h, Y_i)$$

$$y_{n+1} = y_n + \sum_{i=1}^{s} b_i\, hY'_i$$

Method coefficients (A, b, c) are represented in Butcher tableau

One-stage IRK methods

Implicit Euler (order 1)

1	1
	1

Implicit midpoint method (order 2)

$1/2$	$1/2$
	1

$$hY_1' = hf(t_n + c_1 h, y_n + a_{11}\, hY_1')$$
$$y_{n+1} = y_n + b_1\, hY_1'$$

Taylor expansion of $y_{n+1} = y_n + b_1\, hf(t_n + c_1 h, y_n + a_{11}\, hY_1')$

$$y_{n+1} = y + h\, b_1 f + h^2(b_1 c_1 f_t + a_{11} f_y f) + \mathcal{O}(h^3)$$

Taylor expansions for one-stage IRK

Match terms in

$$
\begin{aligned}
y_{n+1} &= y + h\, b_1 f + h^2 (b_1 c_1 f_t + a_{11} f_y f) + \mathcal{O}(h^3) \\
y(t_{n+1}) &= y + h\, f \quad + \frac{h^2}{2} (f_t + f_y f) \qquad\qquad + \mathcal{O}(h^3)
\end{aligned}
$$

Condition for order 1 (consistency) $b_1 = 1$

Condition for order 2 $c_1 = a_{11} = 1/2$

Conclusion Implicit Euler is of order 1 and the implicit midpoint method is the only one-stage 2nd order IRK

4. Stability

Applying an IRK to the linear test equation $y' = \lambda y$, we get

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

Introduce $h\mathbf{Y}' = [hY_1' \cdots hY_s']^{\mathrm{T}}$ and $\mathbf{1} = [1\ 1\ \cdots 1]^{\mathrm{T}} \in \mathbb{R}^s$

Then $(I - h\lambda A)h\mathbf{Y}' = h\lambda\mathbf{1}y_n$ so $h\mathbf{Y}' = h\lambda(I - h\lambda A)^{-1}\mathbf{1}y_n$ and

$$y_{n+1} = y_n + \sum_{j=1}^{s} b_j hY_j' = [1 + h\lambda\mathbf{b}^{\mathrm{T}}(I - h\lambda A)^{-1}\mathbf{1}]y_n$$

The stability function

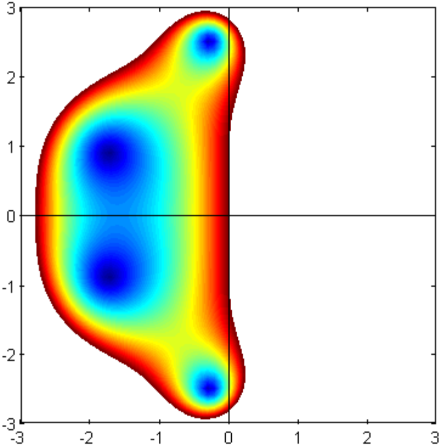
Theorem *For every Runge-Kutta method applied to the linear test equation* $y' = \lambda y$ *we have*

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

where the rational function

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

is called the method's stability function. If the method is explicit, then $R(z)$ *is a polynomial of degree* s


3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3

A-stability of RK methods

Definition *The method's stability region is the set*

$$\mathcal{D} = \{z \in \mathbb{C} : |R(z)| \leq 1\}$$

Theorem *If $R(z)$ maps all of $\mathbb{C}^-$ into the unit circle, then the method is A-stable*

Corollary *No explicit RK method is A-stable*

(For ERK $R(z)$ is a polynomial, and $P(z) \to \infty$ as $z \to \infty$)

A-stability and the Maximum Principle

Theorem *A Runge–Kutta method with stability function $R(z)$ is A-stable if and only if*

- *all poles of R have positive real parts, and*
- $|R(\mathrm{i}\omega)| \leq 1$ *for all* $\omega \in \mathbb{R}$

This is the Maximum Principle in complex analysis

Example

$$\begin{array}{c|cc} 0 & 1/4 & -1/4 \\ 2/3 & 1/4 & 5/12 \\ \hline & 1/4 & 3/4 \end{array} \quad \Rightarrow \quad \begin{array}{l} Y_1' = f(y_n + hY_1'/4 - hY_2'/4) \\ Y_2' = f(y_n + hY_1'/4 + 5hY_2'/12) \\ y_{n+1} = y_n + h(Y_1' + 3Y_2')/4 \end{array}$$

Example. . .

Applied to the test equation, we get the stability function

$$y_{n+1} = \frac{1 + \frac{1}{3}h\lambda}{1 - \frac{2}{3}h\lambda + \frac{1}{6}(h\lambda)^2}\, y_n$$

with poles $2 \pm \mathrm{i}\sqrt{2} \in \mathbb{C}^+$, and

$$|R(\mathrm{i}\omega)|^2 = \frac{1 + \frac{1}{9}\omega^2}{1 + \frac{1}{9}\omega^2 + \frac{1}{36}\omega^4} \le 1$$

Conclusion $|R(h\lambda)| \le 1 \quad \forall\, h\lambda \in \mathbb{C}^-$. The method is *A-stable*

5. Linear Multistep Methods

A *multistep method* is a method of the type

$$y_{n+1} = \Phi(f, h, y_0, y_1, \ldots, y_n)$$

using values from several previous steps

- Explicit Euler $y_{n+1} = y_n + h\,f(t_n, y_n)$
- Trapezoidal rule $y_{n+1} = y_n + h\left(\frac{f(t_n,y_n)+f(t_{n+1},y_{n+1})}{2}\right)$
- Implicit Euler $y_{n+1} = y_n + h\,f(t_{n+1}, y_{n+1})$

are all one-step (RK) methods, but also LM methods

Multistep methods and difference equations

A k-step multistep method replaces the ODE $y' = f(t, y)$ by a *difference equation*

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

Generating polynomials

$$\rho(w) = \sum_{j=0}^{k} a_j\, w^j \qquad\qquad \sigma(w) = \sum_{j=0}^{k} b_j\, w^j$$

- Coefficients are normalized either by $a_k = 1$ or $\sigma(1) = 1$
- $b_k \neq 0 \Leftrightarrow$ *implicit*; $b_k = 0 \Leftrightarrow$ *explicit*

Trivial (one-step) examples

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

$$\rho(w) = w - 1 \qquad \sigma(w) = 1$$

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

$$\rho(w) = w - 1 \qquad \sigma(w) = w$$

Trapezoidal rule $y_{n+1} - y_n = \frac{h}{2}(f(t_{n+1}, y_{n+1}) + f(t_n, y_n))$

$$\rho(w) = w - 1 \qquad \sigma(w) = (w + 1)/2$$

Adams methods (J.C. Adams, 1880s)

Suppose we have the first $n + k$ approximations

$$y_m = y(t_m), \qquad m = 0, 1, \ldots, n + k - 1$$

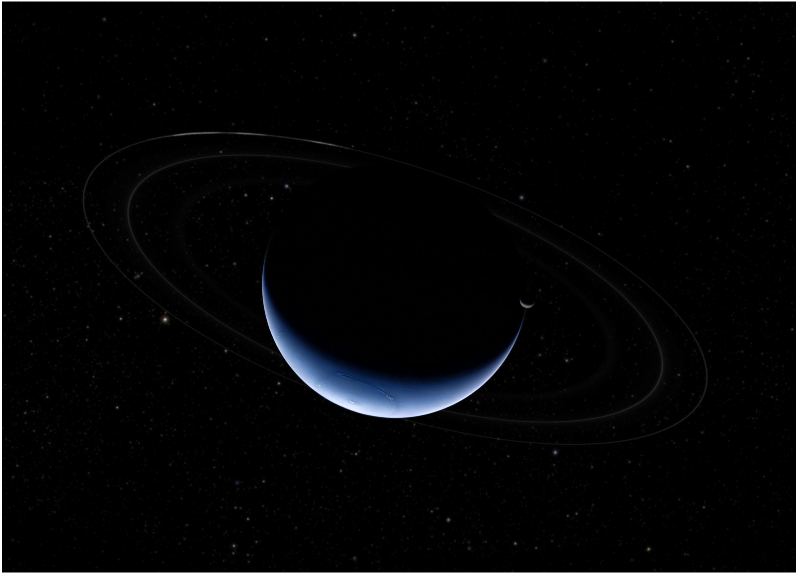
Rewrite $y' = f(t, y)$ by integration

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

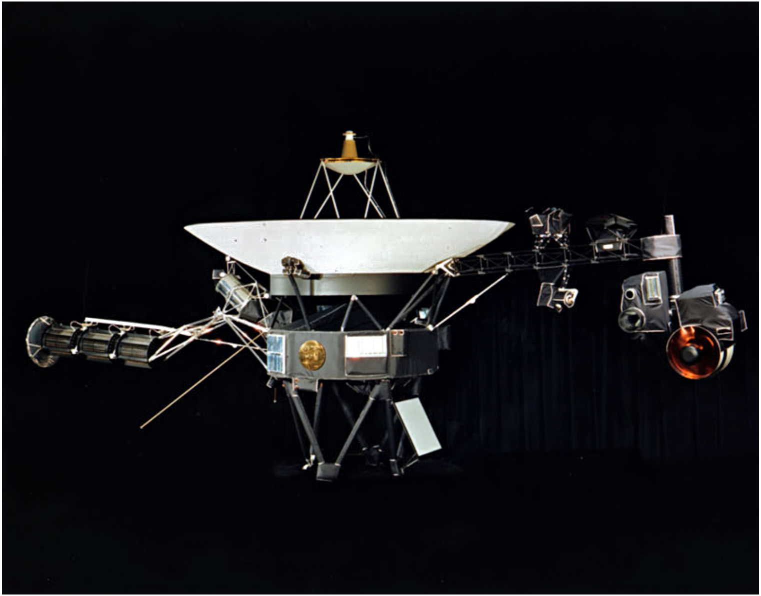
Approximate by an *interpolation polynomial* on $t_n, t_{n-1}, \ldots$

$$f(\tau, y(\tau)) \approx P(\tau)$$

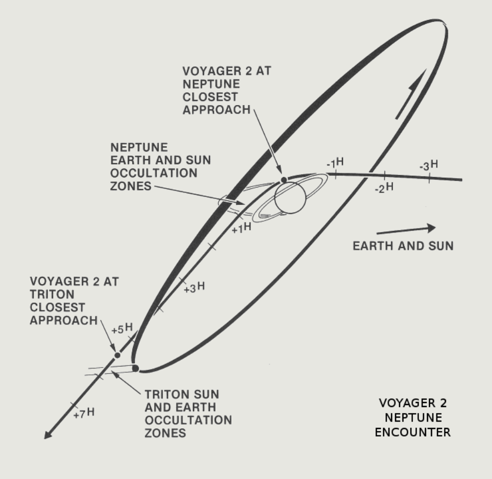
Neptune (1846) *As seen from Voyager 2*



Voyager 2

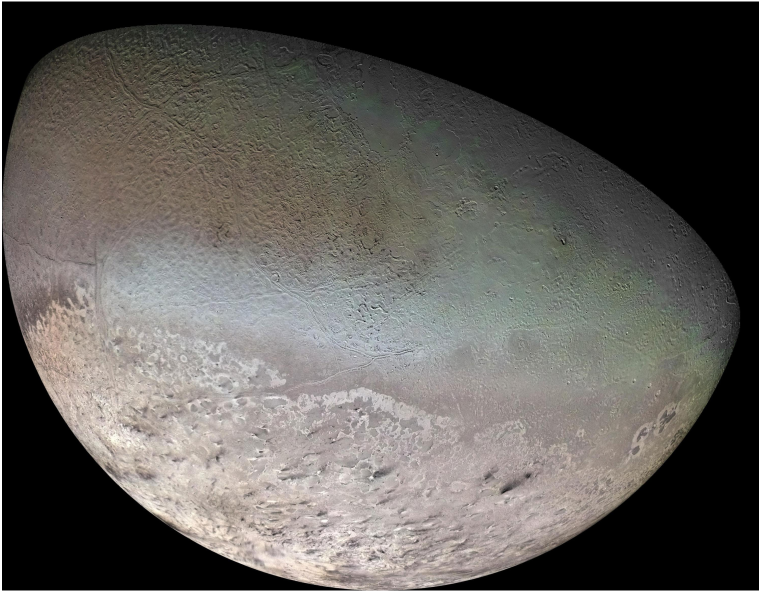


. . . and Adams got the final word 130 years later



Voyager orbit (1977–89) computed using Adams–Moulton methods

Triton, Neptune's moon



Adams–Bashforth methods (explicit)

Approximate $P(\tau) = f(\tau, y(\tau)) + \mathcal{O}(h^k)$, degree $k-1$ polynomial

$$P(t_{n+j}) = f(t_{n+j}, y(t_{n+j})) \qquad j = 0, \dots, k-1$$

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

Adams-Bashforth method (k-step, order $p = k$)

$$y_{n+k} = y_{n+k-1} + \sum_{j=0}^{k-1} b_j\, hf(t_{n+j}, y_{n+j})$$

where $b_j = h^{-1} \int_{t_{n+k-1}}^{t_{n+k}} \varphi_j(\tau)\,\mathrm{d}\tau$ from Lagrange basis polynomials

Coefficients of AB1

For $k = 1$

$$y_{n+1} = y_n + b_0 \, hf(t_n, y_n)$$

the coefficient is determined by

$$b_0 = h^{-1} \int_{t_n}^{t_{n+1}} \varphi_0(\tau) \, \mathrm{d}\tau = h^{-1} \int_{t_n}^{t_{n+1}} 1 \, \mathrm{d}\tau = 1 \quad \Rightarrow$$

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

Conclusion AB1 *is the explicit Euler method*

Coefficients of AB2

For $k = 2$

$$y_{n+2} = y_{n+1} + h\left[b_1\, f(t_{n+1}, y_{n+1}) + b_0\, f(t_n, y_n)\right]$$

with coefficients

$$b_0 = h^{-1} \int_{t_{n+1}}^{t_{n+2}} \frac{\tau - t_{n+1}}{t_n - t_{n+1}}\,\mathrm{d}\tau = -\frac{1}{2}$$
$$b_1 = h^{-1} \int_{t_{n+1}}^{t_{n+2}} \frac{\tau - t_n}{t_{n+1} - t_n}\,\mathrm{d}\tau = \frac{3}{2}$$

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

Initializing an Adams method

The first step of AB2 is

$$y_2 = y_1 + h\left[\frac{3}{2}f(t_1, y_1) - \frac{1}{2}f(t_0, y_0)\right]$$

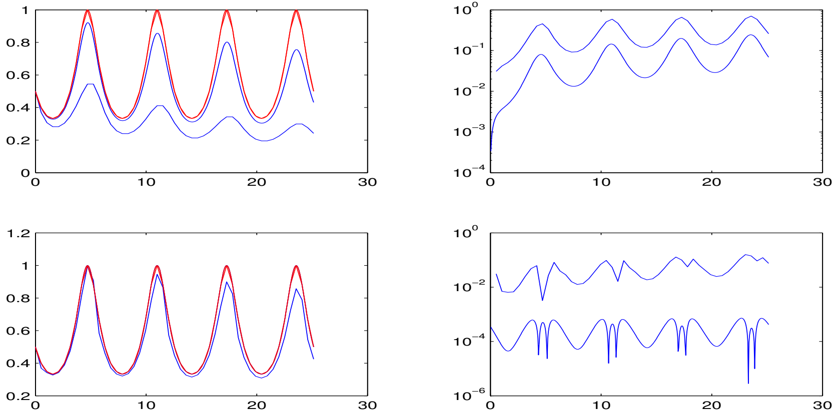
While y_0 is obtained from the initial value, y_1 must be computed with a one-step method, e.g. AB1

$$\begin{aligned} y_1 &= y_0 + h\,f(t_0, y_0) \\ y_{n+2} &= y_{n+1} + h\left[\frac{3}{2}f(t_{n+1}, y_{n+1}) - \frac{1}{2}f(t_n, y_n)\right], \quad n \geq 0 \end{aligned}$$

Multistep software is generally self-starting (with “gearbox”)

Solve $y' = -y^2 \cos t$, $y_0 = 1/2$, $t \in [0, 8\pi]$ using 48 and 480 steps

AB1: Solutions Errors



AB2: Solutions Errors

The order of a multistep method

The *order of consistency* is p if the local error is

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

Taylor expand:

$$Ly = \sum_{j=0}^{k} a_j \Big(y(t_n) + jhy'(t_n) + (jh)^2/2\, y''(t_n) + \cdots \Big) -$$

$$- hb_j \Big(y'(t_n) + jhy''(t_n) + (jh)^2/2\, y^{(3)}(t_n) + \cdots \Big)$$

$$= y(t_n) \left(\sum_{j=0}^{k} a_j \right) + hy'(t_n) \left(\sum_{j=0}^{k} a_j j - b_j \right) + h^2 y''(t_n) \left(\sum_{j=0}^{k} a_j \frac{j^2}{2} - b_j j \right)$$

$$+ \cdots$$

The order of a multistep method, continued

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

Should be $\mathcal{O}(h^{p+1})$ for all solutions y: insert polynomials!

$$y(t) = 1 \implies \sum_{j=0}^{k} a_j = 0$$
$$y(t) = t \implies \sum_{j=0}^{k} a_j j - b_j$$
$$\vdots$$
$$y(t) = t^p \implies \sum_{j=0}^{k} \frac{a_j j^p}{p!} - \frac{b_j j^{p-1}}{(p-1)!}$$

The order of a multistep method, continued

$$\begin{aligned} Ly = y(t_n)\left(\sum_{j=0}^{k} a_j\right) + hy'(t_n)\left(\sum_{j=0}^{k} a_j j - b_j\right) + \\ + h^2 y''(t_n)\left(\sum_{j=0}^{k} \frac{a_j j^2}{2!} - b_j j\right) + h^3 y^{(3)}(t_n)\left(\sum_{j=0}^{k} \frac{a_j j^3}{3!} - \frac{b_j j^2}{2!}\right) \\ + \cdots \end{aligned}$$

Note: $Ly = \mathcal{O}(h^{p+1})$ iff $Ly = 0$ for all polynomials of deg $\leq p$

Easy test for consistency order: insert $y = t^m$ and $y' = mt^{m-1}$

$L(t^q)$ is a polynomial of degree q. If $L(t^q)(jh) = 0$ for all j, then $L(t^q)$ must be the zero polynomial. Thus $L(t^q)(t_n + jh) = 0$ for any t_n. Hence it is enough to check $t_{n+j} = jh$

Order conditions, summary

Theorem *A k-step method is of consistency order p if and only if it satisfies the following conditions*

- $$\sum_{j=0}^{k} j^m a_j = m \sum_{j=0}^{k} j^{m-1} b_j, \quad m = 0, 1, \ldots, p$$
- $$\sum_{j=0}^{k} j^{p+1} a_j \neq (p+1) \sum_{j=0}^{k} j^p b_j$$

A multistep method of consistency order p is *exact* for polynomials of degree $\leq p$. Problems with solutions $y = P(t)$ are solved exactly

Stability

Unlike RK methods there are two distinct kinds of stability notions

- **Finite step stability**

  This is concerned with for what nonzero step sizes h the method can solve the linear test equation $y' = \lambda y$ without going unstable. It determines for what problem classes the method is useful. Same idea as for RK

- **Zero stability**

  This is concerned with whether a multistep method can solve the trivial problem $y' = 0$ without going unstable. If not, the method is useless: *zero stability is necessary for convergence*. Multistep methods only

Definition *A polynomial* $\rho(w)$ *satisfies the root condition if all its zeros are on or inside the unit circle, and the zeros of unit modulus are simple*

Definition *A multistep method whose generating polynomial* $\rho(w)$ *satisfies the root condition is called zero stable*

Examples

- $\rho(w) = (w-1)(w-0.5)$
- $\rho(w) = (w-1)(w+1)$
- $\rho(w) = (w-1)^2(w-0.5)$
- $\rho(w) = (w-1)(w^2+0.25)$

Adams methods have $\rho(w) = w^{k-1}(w-1)$ and are zero stable

The Dahlquist equivalence theorem

Theorem *A multistep method is convergent if and only if it is zero-stable and consistent of order* $p \geq 1$ (without proof)

Example k-step Adams-Bashforth methods are explicit methods of consistency order $p = k$ and have $\rho(w) = w^{k-1}(w-1) \Rightarrow$ they are convergent of *convergence order* $p = k$

Example k-step Adams-Moulton methods are implicit methods of consistency order $p = k+1$ and have $\rho(w) = w^{k-1}(w-1) \Rightarrow$ they are convergent of *convergence order* $p = k+1$

Dahlquist's first barrier theorem

Theorem *The maximal order of a zero-stable k-step method is*

$$p = \begin{cases} k & \text{for explicit methods} \\ \begin{cases} k+1 & \text{if } k \text{ is odd} \\ k+2 & \text{if } k \text{ is even} \end{cases} & \text{for implicit methods} \end{cases}$$

Construct a two-step 2nd order method of the form

$$\alpha_2 y_{n+2} + \alpha_1 y_{n+1} + \alpha_0 y_n = hf(t_{n+2}, y_{n+2})$$

Order conditions for $p = 2$

$$\alpha_2 + \alpha_1 + \alpha_0 = 0; \qquad 2\alpha_2 + \alpha_1 = 1; \qquad 4\alpha_2 + \alpha_1 = 4$$

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

$\rho(w) = \frac{3}{2}(w-1)(w-\frac{1}{3}) \quad \Rightarrow \quad$ BDF2 is convergent of order 2

Backward differentiation formulas (BDF)

Backward difference operator $\nabla y_{n+k} = y_{n+k} - y_{n+k-1}$ with

$$\nabla^j y_{n+k} = \nabla^{j-1} y_{n+k} - \nabla^{j-1} y_{n+k-1}, \quad j > 1$$

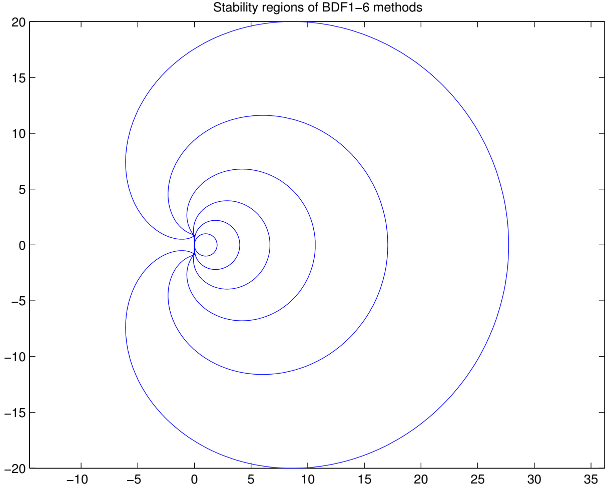
Theorem (without proof) *The k-step BDF method*

$$\sum_{j=1}^{k} \frac{\nabla^j}{j} y_{n+k} = hf(t_{n+k}, y_{n+k})$$

is convergent of order $p = k$ if and only if $1 \leq k \leq 6$

Note BDF methods are designed for *stiff problems*

The methods are stable *outside* the indicated areas



A-stability of multistep methods $y' = \lambda y$

Applying a method to $y' = \lambda y$ produces a difference equation

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

The *characteristic equation* (with $z = h\lambda$)

$$\rho(w) - z\sigma(w) = 0$$

has k roots $w_j(z)$. The method is *A-stable* if and only if

$$\mathrm{Re}\, z \leq 0 \;\Rightarrow\; |w_j(z)| \leq 1,$$

with simple unit modulus roots (root condition)

Dahlquist's second barrier theorem

Theorem (without proof) *The highest order of an A-stable multistep method is* $p = 2$*. Of all 2nd order A-stable multistep methods, the trapezoidal rule has the smallest error*

Note There is no order restriction for Runge–Kutta methods, which can be A-stable for arbitrarily high orders

A multistep method can be useful although it isn't A-stable

6. Difference operators

Differentiation $D : y \mapsto \dot{y}$, where $D = \mathrm{d}/\mathrm{d}t$

Forward shift $E : y(t) \mapsto y(t + h)$

Forward shift applied to sequences

$$y = \{y_n\}_{n=0}^{\infty}$$
$$Ey = \{y_{n+1}\}_{n=0}^{\infty}$$

"Shorthand notation" $(Ey)_n = y_{n+1} \Rightarrow Ey_n = y_{n+1}$

Expand in Taylor series

$$
\begin{aligned}
y(t+h) &= y(t) + h\dot{y}(t) + \frac{h^2}{2}\ddot{y}(t) + \dots \\
&= \left(1 + hD + \frac{(hD)^2}{2!} + \frac{(hD)^3}{3!} + \dots\right) y(t) \\
&= \mathrm{e}^{hD} y(t)
\end{aligned}
$$

Taylor's theorem $E = \mathrm{e}^{hD}$

Forward difference operator

Using short-hand notation

$$\Delta y(t) = y(t+h) - y(t)$$

$$\Delta y_n = y_{n+1} - y_n$$

Note $\Delta = E - 1$

Forward differences of higher order

Recursive definition

$$\Delta y_n = y_{n+1} - y_n$$
$$\Delta^k y_n = \Delta(\Delta^{k-1} y_n)$$

In particular, 2nd order difference

$$\begin{aligned}\Delta^2 y_n &= \Delta(y_{n+1} - y_n) \\ &= (y_{n+2} - y_{n+1}) - (y_{n+1} - y_n) \\ &= y_{n+2} - 2y_{n+1} + y_n\end{aligned}$$

Finite difference approximation of derivatives

Approximation of derivatives

$$\frac{\mathrm{d}y}{\mathrm{d}x} \approx \frac{\Delta y}{\Delta x}$$

$$\frac{\mathrm{d}^2 y}{\mathrm{d}x^2} \approx \frac{\Delta y_n/\Delta x - \Delta y_{n-1}/\Delta x}{\Delta x}$$

$$\approx \frac{y_{n+1} - 2y_n + y_{n-1}}{\Delta x^2} \approx \frac{\Delta^2 y}{\Delta x^2}$$

Backward difference operator

Backward difference

$$\nabla y(t) = y(t) - y(t-h)$$

$$\nabla y_n = y_n - y_{n-1}$$

Bwd Difference $\nabla = 1 - E^{-1}$

Backward shift $E^{-1} = \mathrm{e}^{-hD}$

Linear operators

- All operators under consideration are *linear*

$$L(\alpha u + \beta v) = \alpha L u + \beta L v$$

- Allows *addition* and *multiplication* (assoc + dist laws)
- The operators are *commutative*

$$(L_1 \circ L_2)\, u = (L_2 \circ L_1)\, u$$

- There is a *zero* and a *unit* operator, 0 and 1
- The operators form an *operator algebra*

Operator series and operator calculus

Taylor's theorem

$$e^{-hD} = 1 - \nabla$$

Formal inversion and power series expansion

$$hD = -\log(1 - \nabla) = \nabla + \frac{\nabla^2}{2} + \frac{\nabla^3}{3} + \frac{\nabla^4}{4} + \dots$$

Apply to differential equation $y' = f(y)$

Derivation of BDF methods

Differential equation $y' = f(y)$ implies $hDy = hf(y)$

Replace with *operator series* $hD = \sum_1^\infty \nabla^j/j$. Truncate at k terms

$$\left(\nabla + \frac{\nabla^2}{2} + \cdots + \frac{\nabla^k}{k}\right) y_n = hf(y_n)$$

This is the k-step BDF method

The formula is exact for polynomials of degree $\leq k$, but zero stable only for $k \leq 6$. BDF1–6 are convergent of order $p = k$