1. Runge–Kutta methods
2. Embedded RK methods and adaptivity
3. Implicit Runge–Kutta methods
4. Stability and the stability function
5. Linear multistep methods
6. Difference operators
1. Runge–Kutta methods

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

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

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

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

\[
\begin{array}{c|cccc}
0 & 0 & 0 & \cdots & 0 \\
c_2 & a_{2,1} & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
c_s & a_{s,1} & a_{s,2} & \cdots & 0 \\
\hline
b_1 & b_2 & \cdots & b_s
\end{array}
\]

or

\[
\begin{array}{c|cc}
\mathbf{c} & \mathbf{A} \\
\end{array}
\]

\[
b^T
\]

Simplifying assumption \( c_i = \sum_{j=1}^s a_{i,j} \) (row sums of RK matrix)
A two-stage explicit RK method has \textit{three “free” coefficients}

The simplifying assumption determines the nodes

\[
\begin{align*}
    hY'_1 &= hf(t_n, y_n) \\
    hY'_2 &= hf(t_n + c_2 h, y_n + h a_{21} Y'_1) \\
    y_{n+1} &= y_n + [ b_1 hY'_1 + b_2 hY'_2 ]
\end{align*}
\]

\textbf{Butcher tableau}

\[
\begin{array}{c|ccc}
    0 & 0 & 0 \\
    c_2 & a_{21} & 0 \\
\end{array}
\]

\[
\begin{array}{cc}
    b_1 & b_2 \\
\end{array}
\]

\[c_1 = 0, \; c_2 = a_{21}\]
Derivation of two-stage ERK's

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

\[
hY_2' = hf(t_n + c_2 h, y_n + h a_{21} f(t_n, y_n)) = hf + h^2 [c_2 f_t + a_{21} f_y f] + O(h^3)
\]

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

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

Expand exact solution in Taylor series and match terms

\[
y' = f \\
y'' = f_t + f_y y' = f_t + f_y f \\
y(t + h) = y + hf + \frac{h^2}{2} (f_t + f_y f) + O(h^3)
\]
One-parameter family of 2nd order two-stage ERK’s

Match terms to get conditions for order 2

\[ b_1 + b_2 = 1 \quad \text{(consistency)} \]
\[ b_2 c_2 = 1/2 \]

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

\[
\begin{array}{ccc|ccc}
0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{2b} & \frac{1}{2b} & 0 & \frac{1}{2b} & 0 & 0 \\
1/b & 1/b & 0 & 1/b & 0 & 0 \\
\end{array}
\]
Example 1

The modified Euler method

Put \( b = 1 \) to get the Butcher tableau

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

\[
hY_1' = hf(t_n, y_n)
\]

\[
hY_2' = hf(t_n + h/2, y_n + hY_1'/2)
\]

\[
y_{n+1} = y_n + hY_2'
\]

Second order two-stage explicit Runge–Kutta (ERK) method
Example 2

Heun’s method

Put \( b = 1/2 \) to get

\[
\begin{array}{c|ccc}
0 & 0 & 0 \\
1 & 1 & 0 \\
\hline
1/2 & 1/2
\end{array}
\]

\[
hY_1' = hf(t_n, y_n)
\]

\[
hY_2' = hf(t_n + h, y_n + hY_1')
\]

\[
y_{n+1} = y_n + (hY_1' + hY_2')/2
\]

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

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

\[
\begin{align*}
&b_1 + b_2 + b_3 = 1 \\
&b_2 c_2 + b_3 c_3 = 1/2 \\
&b_2 c_2^2 + b_3 c_3^2 = 1/3 \\
&b_3 a_{32} c_2 = 1/6 
\end{align*}
\]

<table>
<thead>
<tr>
<th>Classical RK3</th>
<th>Nyström scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>1</td>
<td>−1</td>
</tr>
<tr>
<td>1/6</td>
<td>2/3</td>
</tr>
</tbody>
</table>

11 / 61
Exercise

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

\[ hY'_1 = hf(t_n, y_n) \]
\[ hY'_2 = hf(t_n + h/3, y_n + hY'_1/3) \]
\[ hY'_3 = hf(t_n + 2h/3, y_n + 2hY'_2/3) \]

\[ y_{n+1} = y_n + (hY'_1 + 3hY'_3)/4 \]

Is the method of order 3?
Classical RK4

4th order, 4-stage ERK

The “original” RK method (1895)

\[ hY_1' = hf(t_n, y_n) \]
\[ hY_2' = hf(t_n + h/2, y_n + hY_1'/2) \]
\[ hY_3' = hf(t_n + h/2, y_n + hY_2'/2) \]
\[ hY_4' = hf(t_n + h, y_n + hY_3') \]

\[ y_{n+1} = y_n + \frac{1}{6} (hY_1' + 2hY_2' + 2hY_3' + hY_4') \]
Classical RK4 . . .

Butcher tableau

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1/2</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1/2</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1/6</td>
<td>1/3</td>
<td>1/3</td>
<td>1/6</td>
<td></td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>stages $s$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>coefficients</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>15</td>
<td>21</td>
<td>28</td>
<td>36</td>
<td>45</td>
<td>55</td>
<td>66</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>order $p$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>conditions</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>17</td>
<td>37</td>
<td>85</td>
<td>200</td>
<td>486</td>
<td>1205</td>
<td></td>
</tr>
<tr>
<td>min stages</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
2. Embedded RK methods

Two methods in a single Butcher tableau (RK34)

\[ hY_1' = hf(t_n, y_n) \]
\[ hY_2' = hf(t_n + h/2, y_n + hY_1'/2) \]
\[ hY_3' = hf(t_n + h/2, y_n + hY_2'/2) \]
\[ hZ_3' = hf(t_n + h, y_n - hY_1' + 2hY_2') \]
\[ hY_4' = hf(t_n + h, y_n + hY_3') \]

\[ y_{n+1} = y_n + \frac{1}{6} (hY_1' + 2hY_2' + 2hY_3' + hY_4') \quad \text{order 4} \]
\[ z_{n+1} = y_n + \frac{1}{6} (hY_1' + 4hY_2' + hZ_3') \quad \text{order 3} \]

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

Adjust the step size $h$ to make local error estimate equal to a prescribed *error tolerance* $\text{TOL}$

Simplest step size updating scheme

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

makes $r_n \approx \text{TOL}$

Time step adaptivity using local error control
Advanced adaptive RK methods

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

- Dormand–Prince DOPRI45
- Dormand–Prince DOPRI78
- Cash–Karp CK5

Advanced adaptivity uses discrete control theory and digital filters

\[ h_{n+1} = \rho_n \cdot h_n \]

\[ \rho_n = \left( \frac{TOL}{r_{n+1}} \right)^{\beta_1/p} \left( \frac{TOL}{r_n} \right)^{\beta_2/p} \rho_{n-1}^{-\alpha} \]

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

\[
hY'_i = hf(t_n + c_i h, y_n + \sum_{j=1}^{s} a_{i,j} hY'_j)
\]

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

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)  Implicit midpoint method (order 2)

\[
\begin{array}{c|c|c|c|c|c}
0 & 1 & \frac{1}{2} & 0 & 1 \\
1 & 1 & 1 & 1 & 1
\end{array}
\]

\[
hY'_1 = hf(t_n + c_1 h, y_n + a_{11} hY'_1)
\]
\[
y_{n+1} = y_n + b_1 hY'_1
\]

Taylor expansion of \( y_{n+1} = y_n + b_1 hf(t_n + c_1 h, y_n + a_{11} hY'_1) \)

\[
y_{n+1} = y + h b_1 f + h^2 (b_1 c_1 f_t + a_{11} f_y f) + O(h^3)
\]
Taylor expansions for one-stage IRK

Match terms in

\[ y_{n+1} = y + h b_1 f + h^2 (b_1 c_1 f_t + a_{11} f_y f) + O(h^3) \]
\[ y(t_{n+1}) = y + h f + \frac{h^2}{2} (f_t + f_y f) + O(h^3) \]

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 $hY' = [hY'_1 \cdots hY'_s]^T$ and $1 = [1 \ 1 \cdots 1]^T \in \mathbb{R}^s$

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

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

\[
R(z) = 1 + zb^T(I --zA)^{-1}1
\]

is called the method’s **stability function**. If the method is explicit, then \( R(z) \) is a polynomial of degree \( s \)
RK4 stability region

Contours of $|R(z)|$
A-stability of RK methods

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

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

**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
- $|R(i\omega)| \leq 1$ for all $\omega \in \mathbb{R}$

This is the Maximum Principle in complex analysis

**Example**

<table>
<thead>
<tr>
<th></th>
<th>1/4</th>
<th>−1/4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2/3</td>
<td>1/4</td>
<td>5/12</td>
</tr>
<tr>
<td></td>
<td>1/4</td>
<td>3/4</td>
</tr>
</tbody>
</table>

\[
Y'_1 = f(y_n + hY'_1/4 - hY'_2/4) \quad \Rightarrow \quad Y'_2 = f(y_n + hY'_1/4 + 5hY'_2/12) \quad \Rightarrow \quad y_{n+1} = y_n + h(Y'_1 + 3Y'_2)/4
\]
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 $2 \pm i \sqrt{2} \in \mathbb{C}^+$, and

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

**Conclusion**

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

\[
\rho(w) = \sum_{j=0}^{k} a_j w^j \quad \sigma(w) = \sum_{j=0}^{k} b_j w^j
\]

- Coefficients are normalized either by \( a_k = 1 \) or \( \sigma(1) = 1 \)
- \( b_k \neq 0 \iff \text{implicit} \); \( b_k = 0 \iff \text{explicit} \)
Trivial (one-step) examples

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

\[ \rho(w) = w - 1 \quad \sigma(w) = 1 \]

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

\[ \rho(w) = w - 1 \quad \sigma(w) = w \]

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

\[ \rho(w) = w - 1 \quad \sigma(w) = (w + 1)/2 \]
Adams methods (J.C. Adams, 1880s)

Suppose we have the first $n + k$ approximations

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

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

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

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

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

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

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

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

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

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

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

**Conclusion**  AB1 *is the explicit Euler method*
Coefficients of AB2

For $k = 2$

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

with coefficients

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

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

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

\[ y_1 = y_0 + hf(t_0, y_0) \]

\[ y_{n+2} = y_{n+1} + h \left[ \frac{3}{2} f(t_{n+1}, y_{n+1}) - \frac{1}{2} f(t_n, y_n) \right], \quad n \geq 0 \]

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

40 / 61
A computational comparison

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

AB1: Solutions

AB2: Solutions

AB1: Errors

AB2: Errors

AB1 vs AB2
How do we check the order of a multistep method?

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

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

Try if the formula holds exactly for polynomials

$$y(t) = 1, \quad t, \quad t^2, \quad t^3, \ldots$$

Insert $y = t^m$ and $y' = mt^{m-1}$ into the formula, taking $t_{n+j} = jh$

$$\sum_{j=0}^{k} a_j (jh)^m - h \sum_{j=0}^{k} b_j m(jh)^{m-1} = h^m \sum_{j=0}^{k} (a_j j^m - b_j m j^{m-1})$$
Theorem  A $k$-step method is of consistency order $p$ if and only if it satisfies the following conditions

\begin{align*}
\text{• } \sum_{j=0}^{k} j^m a_j &= m \sum_{j=0}^{k} j^{m-1} b_j, \quad m = 0, 1, \ldots, p \\
\text{• } \sum_{j=0}^{k} j^{p+1} a_j &\neq (p + 1) \sum_{j=0}^{k} j^p b_j
\end{align*}

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
Zero stability

**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 \textit{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 \textit{convergence order} \( p = k + 1 \)
Theorem  The maximal order of a zero-stable $k$-step method is

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

for implicit methods
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 \( p = 2 \)

\[ \alpha_2 + \alpha_1 + \alpha_0 = 0; \quad 2\alpha_2 + \alpha_1 = 1; \quad 4\alpha_2 + \alpha_1 = 4 \]

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

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

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

$$
\Re z \leq 0 \Rightarrow |w_j(z)| \leq 1,
$$

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 \( D = \frac{d}{dt} \)

**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”  \( (Ey)_n = y_{n+1} \Rightarrow Ey_n = y_{n+1} \)
Taylor’s theorem

Expand in Taylor series

\[ y(t + h) = y(t) + h\dot{y}(t) + \frac{h^2}{2} \ddot{y}(t) + \ldots \]
\[ = \left( 1 + hD + \frac{(hD)^2}{2!} + \frac{(hD)^3}{3!} + \ldots \right) y(t) \]
\[ = e^{hD} y(t) \]

Taylor’s theorem \( E = e^{hD} \)
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

\[ \Delta^2 y_n = \Delta(y_{n+1} - y_n) \]
\[ = (y_{n+2} - y_{n+1}) - (y_{n+1} - y_n) \]
\[ = y_{n+2} - 2y_{n+1} + y_n \]
Finite difference approximation of derivatives

Approximation of derivatives

\[
\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x}
\]

\[
\frac{d^2y}{dx^2} \approx \frac{\Delta y_n/\Delta x - \Delta y_{n-1}/\Delta x}{\Delta x}
\]

\[
\approx \frac{y_{n+1} - 2y_n + y_{n-1}}{\Delta x^2} \approx \frac{\Delta^2 y}{\Delta x^2}
\]
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** \( E^{-1} = e^{-hD} \)
Linear operators

- All operators under consideration are \textit{linear}

\[ L(\alpha u + \beta v) = \alpha Lu + \beta Lv \]

- Allows \textit{addition} and \textit{multiplication} (assoc + dist laws)
- The operators are \textit{commutative}

\[ (L_1 \circ L_2) u = (L_2 \circ L_1) u \]

- There is a \textit{zero} and a \textit{unit} operator, \textit{0} and \textit{1}
- The operators form an \textit{operator algebra}
Taylor’s theorem
\[ e^{-hD} = 1 - \nabla \]

Formal inversion and power series expansion
\[ hD = -\log(1 - \nabla) = \nabla + \frac{\nabla^2}{2} + \frac{\nabla^3}{3} + \frac{\nabla^4}{4} + \ldots \]

Apply to differential equation \( y' = f(y) \)
Derivation of BDF methods

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

Replace with operator series $hD = \sum_{j=1}^{\infty} \nabla^j / j$. 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 $p = k$