Topic and contents

UNSW, School of Mathematics and Statistics

MATH2089 – Numerical Methods

Week 8 – Ordinary Differential Equations II

- MATLAB M-files
  - eulerf.m
  - heun.m
  - rk4.m
  - ivpmain.m
  - bvpex1.m

Implicit Euler and Heun Methods

- IVP: $y' = f(t, y), \quad y(t_0) = y_0$
- Explicit Euler $y_{n+1} = y_n + hf(t_n, y_n)$
  - MATLAB function eulerf.m
- Implicit Euler's method
  - Approximation $f(t, y) \approx f(t_{n+1}, y_{n+1})$ on $[t_n, t_{n+1}]$

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}), \quad n = 0, 1, \ldots, N-1$$

  - Implicit $\iff$ require solution of (nonlinear) equation to get y_{n+1}
- Heun's method: For $n = 0, 1, \ldots, N-1$

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

  - Example of a predictor-corrector method
  - Prediction $\bar{y}_{n+1} = y_n + hf(t_n, y_n)$
  - Correction $y_{n+1} = \frac{h}{2}\left[f(t_n, y_n) + f(t_{n+1}, \bar{y}_{n+1})\right]$ uses prediction
  - MATLAB function heunf.m

Local vs Global Error

- Local Truncation Error

$$T(t) = \frac{y(t+h) - y(t)}{h} - f(t, y(t))$$

  - Truncation error at t_n is $T_n = T(t_n)$
  - Euler's method $T_n = O(h)$
  - MATLAB ivpmain.m, Example 2

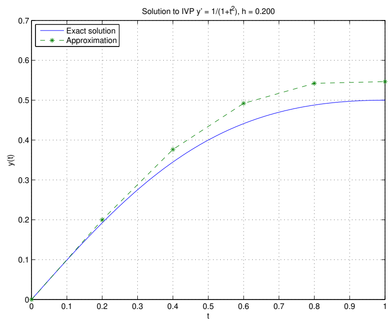
- Global error

$$E_n(h) = y(t_n) - y_n$$

  - Convergence

$$\lim_{h \to 0} \max_n |E_n(h)| = 0$$

  - Euler's method $E_n = O(h)$
    slow



Explicit Runge-Kutta methods

Definition (Explicit Runge-Kutta (ERK) Methods)

A ν-stage explicit Runge-Kutta method with parameters a_{ij}, b_j, c_j is

$$\begin{aligned}
\xi_1 &= y_n \\
\xi_2 &= y_n + h a_{2,1} f(t_n + c_1 h, \xi_1) \\
\xi_3 &= y_n + h a_{3,1} f(t_n + c_1 h, \xi_1) + h a_{3,2} f(t_n + c_2 h, \xi_2) \\
&\vdots \\
\xi_\nu &= y_n + h \sum_{i=1}^{\nu-1} a_{\nu,i} f(t_n + c_i h, \xi_i)
\end{aligned}$$

Then

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

Runge-Kutta Parameters

- RK matrix $A \in \mathbb{R}^{\nu\times\nu}$
  - A is strictly lower triangular, eg $A_{ij} = 0$ for $j \geq i$
- RK weights $\mathbf{b} = (b_1,\ b_2,\ \ldots, b_\nu)^T \in \mathbb{R}^\nu$
- RK nodes $\mathbf{c} = (c_1,\ c_2,\ \ldots, c_\nu)^T \in \mathbb{R}^\nu, \quad c_1 = 0$
- Parameters A, $\mathbf{b}$, $\mathbf{c}$ displayed in RK tableau

$$\begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array}$$

- Two stage RK methods, $E_n = O(h^2)$, (Heun is third)

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array} \qquad \begin{array}{c|cc} 0 & & \\ \frac{2}{3} & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array} \qquad \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Three and four stage Runge-Kutta Methods

- Three stage RK methods, $E_n = O(h^3)$ (Classical RK method, Nyström)

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array} \qquad \begin{array}{c|ccc} 0 & & & \\ \frac{2}{3} & \frac{2}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{8} & \frac{3}{8} \end{array}$$

- Four stage RK method $E_n = O(h^4)$ (RK4)

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

  - MATLAB function `rk4.m`

Runge-Kutta Method – Example

Example

Write down the formulae for the Runge-Kutta method with the RK tableau

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

Use this method with $h = 0.25$ to estimate $y(1.5)$ for the IVP

$$y' = 1 + \frac{y}{t}, \qquad y(1) = 2$$

Solution

- *RK tableau gives*

$$\begin{aligned}
\xi_1 &= y_n \\
\xi_2 &= y_n + \frac{2}{3} h f(t_n, \xi_1) \\
y_{n+1} &= y_n + h\left[\frac{1}{4} f(t_n, \xi_1) + \frac{3}{4} f(t_n + \frac{2}{3}h, \xi_2)\right]
\end{aligned}$$

Runge-Kutta Method – Example

Solution

- $f(t, y) = 1 + \frac{y}{t}$, $\quad t_0 = 1$, $\quad y_0 = 2$, $\quad h = 0.25$
-

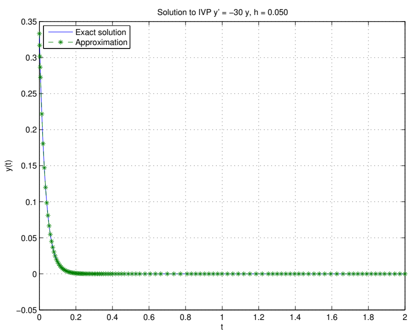
n	t_n	$\xi_1 = y_n$	$f(t_n, \xi_1)$	ξ_2	$f(t_n + \frac{2}{3}h, \xi_2)$
0	1	2	3	2.5	3.1429
1	1.25	2.7768	3.2214	3.3137	3.3391
2	1.5	3.6042			

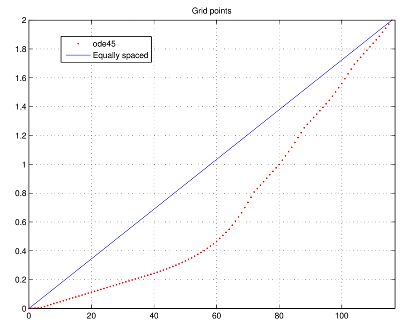
Step Size Control

- Adjust step-size to keep local error estimate within tolerance
- Interval halving: at t_n, y_n
  - Use one step of h to get $y_{n+1}(h)$
  - Use two steps of $h/2$ to get $y_{n+1}(h/2)$
  - Local truncation error of method gives estimate of T_{n+1}
  - Reduce step until error estimate within desired tolerance
  - $T(h) = O(h^5) \implies$ halving h reduces error by $1/2^5 = 1/32$
  - To reduce error by 10 need to reduce h by $10^{1/5} \approx 1.58$
- Runge-Kutta-Fehlberg Method
  - Use difference between different order RK methods to give estimate of local truncation error
  - Runge-Kutta-Fehlberg Method uses fourth and fifth order methods
  - Fewer function evaluations than interval halving
  - MATLAB function ode23, ode45

Step-Size Control – ODE45

- $y' = -30y$, $\quad y(0) = \frac{1}{3}$, using MATLAB ode45





Multi-Step Methods

Definition

A multi-step method uses r previous values y_k for $k \le n$ to determine

$$y_{n+1} = y_n + h \sum_{j=-1}^{r} b_j f(t_{n-j}, y_{n-j}), \quad n \ge r$$

- $b_{-1} = 0 \implies$ explicit method
- $b_{-1} \ne 0 \implies$ implicit method (y_{n+1} on both sides, solve equation)
- $b_r \ne 0 \implies r+1$ step method using $y_{n-r}, y_{n-r+1}, \ldots, y_n$
- Need r starting values $y_1, \ldots, y_r$ (eg from RK method)
- $r = 0$ one-step methods

Predictor-Corrector Methods

- Notation $f_n = f(t_n, y_n)$, $\quad f_{n+1} = f(t_{n+1}, y_{n+1})$, etc
- Adams-Bashforth Predictor (AB4)

$$y_{n+1} = y_n + \frac{h}{24}(55 f_n - 59 f_{n-1} + 37 f_{n-2} - 9 f_{n-3})$$

  - Local truncation error $O(h^5)$, Global truncation error $O(h^4)$
- Adams-Moulton Corrector (AM4)

$$y_{n+1} = y_n + \frac{h}{24}(9 f_{n+1} + 19 f_n - 5 f_{n-1} + f_{n-2})$$

  - y_{n+1} from predictor to get f_{n+1}
  - Local truncation error $O(h^5)$, Global truncation error $O(h^4)$

Stiff Problems

An IVP is stiff when very small step sizes may be required for an explicit method to get an accurate solution.

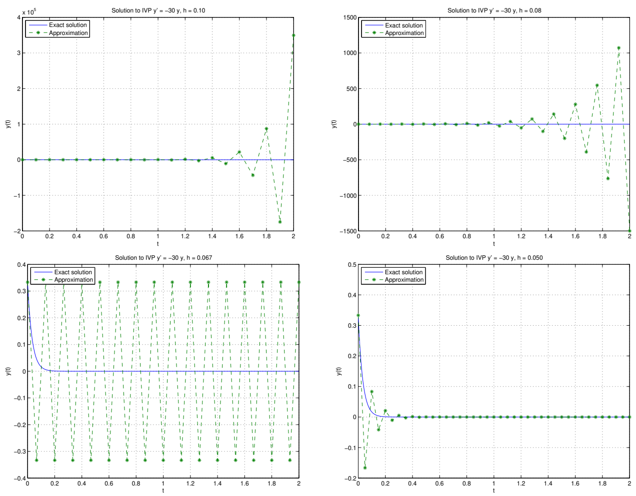
Example (Stiff IVP)

Consider the IVP $y' = -30y$ for $t \ge 0$ with initial value $y(0) = 1/3$. Solve on $[0, 2]$

- Solution $y(t) = \frac{1}{3} e^{-30t}$
- As $t \to \infty$, $y(t) \to 0$
- Explicit methods only have this behaviour for small h
- Implicit methods much better for stiff problems
- MATLAB `ivpmain.m`, Example 6 with Euler, Heun, RK4

Stiff Problem – Euler's Method

- Euler's method for $N = 20, 25, 30, 40$
- Corresponding $h = 0.1, 0.08, 0.06666, 0.05$



Euler's Method – Stability

Example

Test problem $y' = cy$ with $y(0) = a$

- Solution $y(t) = ae^{ct}$
- $c < 0 \implies y(t) \to 0$ as $t \to \infty$
- Euler:

$$\begin{aligned} y_n &= y_{n-1} + hf(t_{n-1}, y_{n-1}) \\ &= y_{n-1} + hcy_{n-1} = (1+ch)y_{n-1} \\ &= (1+ch)^2 y_{n-2} \\ &\vdots \\ &= (1+ch)^n y_0 \end{aligned}$$

- $y_n = (1+ch)^n y_0$ diverges unless $|1+ch| < 1$
    - $c < 0 \implies h < \frac{2}{|c|}$
    - Example $c = -30 \implies h < 1/15 = 0.0666$

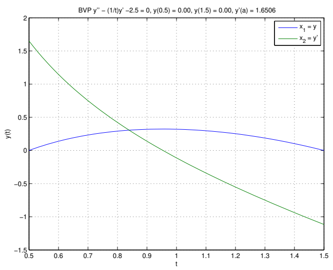
Boundary Value Problems (BVP)

- Second order differential equation $y'' = g(t, y, y')$ for $t \in [a, b]$
- Two-dimensional first-order system: state vector $\mathbf{x} = [y,\ y']^T$
- Initial value problem $\mathbf{x}(a) = (y(a),\ y'(a))^T$
- Boundary value problem $y(a) = y_a$ and $y(b) = y_b$

Example

BVP $y'' + \frac{1}{t}y' + 2.5 = 0$, on $[0.5, 1.5]$ with $y(0.5) = 0$ and $y(1.5) = 0$

- $\mathbf{x} = [y,\ y']^T$
- $\mathbf{x}' = [y',\ g(t, y, y')]^T$
- $g(t, y, y') = -\frac{1}{t}y' - 2.5$



BVP – Shooting Methods

- Shooting Methods
    - First order system

$$\mathbf{x} = \begin{bmatrix} y \\ y' \end{bmatrix}, \quad \mathbf{x}' = \begin{bmatrix} y' \\ g(x, y, y') \end{bmatrix}, \quad \mathbf{x}(a) = \begin{bmatrix} y_a \\ \eta \end{bmatrix}$$

    - Initial conditions with a parameter $\eta \in \mathbb{R}$
    - Solve IVP to get $y(b; \eta)$
    - Choose parameter η: $y(b; \eta) = y_b \iff \psi(\eta) = y(b; \eta) - y_b = 0$
    - Solve $\psi(\eta) = 0$
        - Iterative method (fixed point iteration, Newton, Secant)
        - $\implies$ new estimate of η
        - Re-solve IVP with new initial conditions
        - MATLAB `bvpex1.m`