Explicit Runge-Kutta methods

$\boxed{\text{ode2}}$

Explicit vs Implicit

$$\left.\begin{array}{l} \dfrac{d}{dt}\begin{pmatrix} \theta \\ \omega \end{pmatrix} = \begin{pmatrix} \omega \\ -\gamma\omega - (g/\ell + A\cos 2\pi t)\sin\theta \end{pmatrix} \\ \\ \theta(0) = \theta_0 \;, \; \omega(0) = \omega_0 \end{array}\right\} \text{IVP}$$

Forward Euler

$$\begin{pmatrix} \theta_{k+1} \\ \omega_{k+1} \end{pmatrix} = \begin{pmatrix} \theta_k \\ \omega_k \end{pmatrix} + h \begin{pmatrix} \omega_k \\ -\gamma\omega_k - (g/\ell + A\cos 2\pi t_k)\sin\theta_k \end{pmatrix}$$

$$\theta_{k+1} = \theta_k + h\omega_k$$

$$\omega_{k+1} = (1-\gamma h)\,\omega_k - h\left(g/l + A\cos 2\pi t_k\right)\sin\theta_k$$

Explicit "marching" scheme

Backward Euler

$$\begin{pmatrix} \theta_{k+1} \\ \omega_{k+1} \end{pmatrix} = \begin{pmatrix} \theta_k \\ \omega_k \end{pmatrix} + h\begin{pmatrix} \omega_{k+1} \\ -\gamma\omega_{k+1} - \left(\frac{g}{l} + A\cos 2\pi t_{k+1}\right)\sin\theta_{k+1} \end{pmatrix}$$

$$\theta_{k+1} - h\omega_{k+1} = \theta_k$$

$$(1+h\gamma)\,\omega_{k+1} + \left(\frac{g}{l} + A\cos 2\pi t_{k+1}\right)\sin\theta_{k+1} = \omega_k$$

Implicitly define $\theta_{k+1}, \omega_{k+1}$ (Root finding)

Butcher Tables

$$\begin{array}{c|c} \vec{c} & A \\ \hline & \vec{b} \end{array} = \begin{array}{c|cccccc} c_1 & 0 & & & & & \\ c_2 & a_{21} & 0 & & & & \\ c_3 & a_{31} & a_{32} & 0 & \ddots & & \\ \vdots & \vdots & & & & & \\ c_s & a_{s1} & a_{s2} & a_{s3} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & b_3 & \cdots & b_{s-1} & b_s \end{array}$$

$\vec{c}$ stage times

$\vec{b}$ stage expansion

A Runge-Kutta matrix

$$\boxed{\text{have } (t_n, y_n) \text{, get } (t_{n+1}, y_{n+1})}$$

Example

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

$$k_1 = f(t_n, y_n)$$
$$k_2 = f(t_n + h, y_n + h k_1)$$
$$y_{n+1} = y_n + \frac{h}{2}(k_1 + k_2)$$

explicit trapezoid

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

$$k_1 = f(t_n, y_n)$$
$$k_2 = f(t_n + \tfrac{1}{2}h, y_n + \tfrac{1}{2}h k_1)$$
$$y_{n+1} = y_n + h k_2$$

explicit midpoint

Both RK2.

Example

$$\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\
\frac{2}{3} & -\frac{1}{3} & 1 & 0 & 0 \\
1 & 1 & -1 & 1 & 0 \\
\hline
 & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8}
\end{array}$$

$$k_1 = f(t_n, y_n)$$

$$k_2 = f\left(t_n + \frac{h}{3}, y_n + \frac{h}{3} k_1\right)$$

$$k_3 = f\left(t_n + \frac{2h}{3}, y_n - \frac{h}{3} k_1 + h k_2\right)$$

$$k_4 = f(t_n + h, y_n + h k_1 - h k_2 + h k_3)$$

$$y_{n+1} = y_n + \frac{h}{8}(k_1 + 3k_2 + 3k_3 + k_4)$$

Classical RK4

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

?

Embedded methods and Error control

ODE 2

EX

$$\begin{array}{c|c} \vec{c} & A \\ \hline & \vec{b} \\ & \vec{b}' \end{array} = \begin{array}{c|cc} 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \\ & 1 & 0 \end{array}$$

$$k_1 = f(t_n, y_n)$$

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

Two updates: $y_n + h k_2$, $y_n + h k_1$

Compare for error control

function $[t, y, h, nsteps] = \text{TakeStep}(f, t_0, y_0, h_0, nstep_0)$

$k_1 = f(t_0, y_0)$

$k_2 = f(t_0 + \frac{1}{2}h_0,\ y_0 + \frac{h_0}{2} k_1)$

$\delta = \frac{1}{3} h_0 \max\left(\text{abs}(k_2 - k_1)./\text{abs}(y_0)\right)$

if $(\delta < tol)$

$y = y_0 + h_0 k_2$

$t = t_0 + h_0$

$h = h_0 * \min(1.5, \sqrt{tol/(1.2\delta)})$

$nsteps = nsteps_0 + 1$

else

$y = y_0$, $t = t_0$, $nsteps = nsteps_0$

$h = h_0 * \min(0.1, \sqrt{tol/(1.2\delta)})$

end

Nguess = 10,000 (guess at max # of steps)

```
g = 1, t = 0, y = y0, h = 0.025, tol = 5e-4
while (t < Maxtime)
    nsteps0 = nsteps
    [t, y, h, nsteps] = TakeStep(@f, t, y, h, nsteps, tol)
    if nsteps > nstep0
        g = g + 1
        if (g > Nguess + 1) stop end
    end
    if (Maxtime - t < h) h = maxtime - t end
end
```