

Lecture 29

Numerical ODEs: Runge-Kutta

L. Olson

Department of Computer Science
University of Illinois at Urbana-Champaign

April 27, 2006

Course Timeline

- Th 4/27: Runge-Kutta, ode45, Wiki final draft due
- Tu 5/2: ODE Review, Comprehensive Review
- Th 5/4: HW11 due (no late accepted), Review
- Fr 5/5: Final exam, 7-10pm, 1404-SC, Wiki closed

Wiki of the day

Highlight some interesting progress:

Cheap Matrix Factorizations with ILU

- One way to make Gaussian Elimination faster is to ignore “fill-in” values that are small
- this is Incomplete LU factorization (ILU)
- +5 extra wiki points

ODE Recall

$$\frac{dy}{dt} = f(t, y)$$

Euler's Method

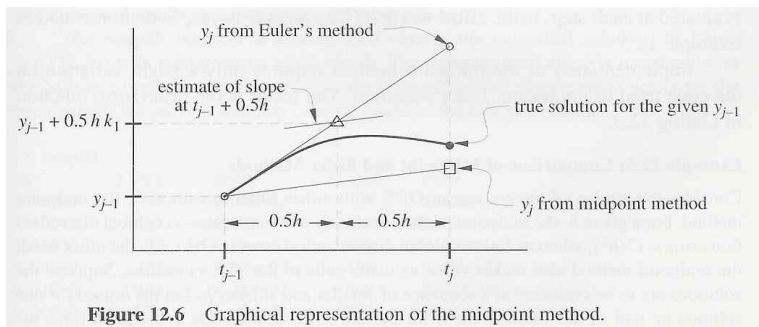
$$y_j = y_{j-1} + hf(t_{j-1}, y_{j-1})$$

with initial conditions $y_0 = y(t_0)$

- Based on the slope at the beginning of the interval
- $\mathcal{O}(h)$ LDE and GDE

To Midpoint

- Euler: Use forward differences to approximate y'
- Midpoint: Use central differences to approximate y'



To Midpoint

Midpoint Method

$$y_j = y_{j-1} + hf\left(t_{j-1} + \frac{h}{2}, y_{j-1} + \frac{h}{2}f\left(t_{j-1}, y_{j-1}\right)\right)$$

- Based on the slope in the *middle* of the interval
- $\mathcal{O}(h^2)$ LDE and GDE

Question: Can we improve cheaply?

- Midpoint used 2 function evaluations. Can we use two function evaluation in a better way?
- Let's base the approximation on two slopes:
 - 1 First take the slope at the beginning of the interval. This we know.
 - 2 Next, use this slope to help approximate the slope at the end of the interval

Heun's Method

- slope at the beginning of the interval:

$$k_1 = f(t_{j-1}, y_{j-1})$$

- use k_1 to estimate the end-slope k_2 :

$$k_2 = f(t_{j-1} + h, y_{j-1} + hk_1)$$

- Then average:

Heun's Method

$$y_j = y_{j-1} + h \frac{k_1 + k_2}{2}$$

Heun's Method

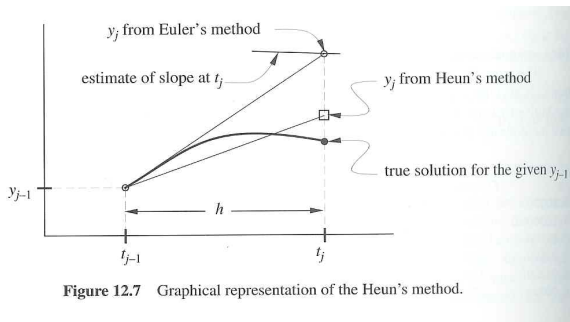


Figure 12.7 Graphical representation of the Heun's method.

- $\mathcal{O}(h^2)$ LDE and GDE (same as Midpoint)
- same cost as Midpoint

RK4

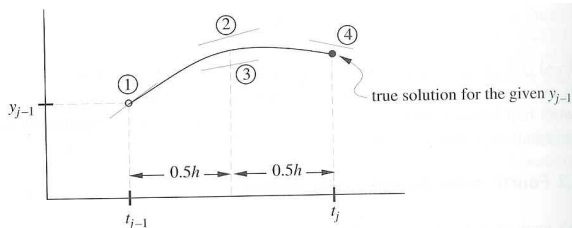


Figure 12.8 Graphical representation of the four points at which the slope is evaluated in fourth-order Runge–Kutta method.

- Question: if we use *more* slopes, does this improve accuracy?
- This is Runge-Kutta with fancy averaging:

$$y_j = y_{j-1} + \sum_{\ell=1}^m \gamma_{\ell} K_{\ell}$$

- Heun's Method ($m = 2$): $\gamma_1 = \gamma_2 = 0.5$
- One requirement for the average: $\sum_{\ell=1}^m \gamma_{\ell} = 1$

RK4

- To obtain a GDE of $\mathcal{O}(h^4)$, use 4 estimates:

$$k_1 = f(t_{j-1}, y_{j-1})$$

$$k_2 = f\left(t_{j-1} + \frac{h}{2}, y_{j-1} + \frac{h}{2}k_1\right)$$

$$k_3 = f\left(t_{j-1} + \frac{h}{2}, y_{j-1} + \frac{h}{2}k_2\right)$$

$$k_4 = f(t_{j-1} + h, y_{j-1} + hk_3)$$

- Use a weighted average

RK4

$$y_j = y_{j-1} + h \left(\frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \right)$$

ode45, ode23, ode23s

- Matlab's ode45 and ode23 are *adaptive* RK methods:
Runge-Kutta-Fehlberg methods
- The adaptive methods have a target tolerance for the GDE
- The method will speed up or slow down in order to satisfy this tolerance
- Last HW question deals with ode45
- ode23s is for “stiff” ODE. These are problematic with standard approaches
- What to get out of the last problem:
 - 1 Visualization of adaptive RK trying to be accurate (oscillations near the step)
 - 2 The robustness of methods built specifically for stiff problems “ode23s”