Math 211

Lecture #13

Runge-Kutta Methods

February 14, 2001

Experimental Error Analysis

- IVP $y' = \cos(t)/(2y-2)$ with $y(0) = 3$
- Exact solution: $y(t) = 1 + \sqrt{4 + \sin t}$.
- Solve using Euler's method and compare with the exact solution.
- Do this for several step sizes.

Derivative m-file ben.m

```
function yprime = ben(t,y)

yprime = cos(t)/(2*y-2);
```

M-file batch.m

```
[teuler,yeuler]=eul('ben',[0,3],3,h);
t=0:0.05:3;
y=1+sqrt(4+sin(t));
plot(t,y,teuler,yeuler,'o')
legend('Exact','Euler')
shg
z=1+sqrt(4+sin(teuler));
maxerror=max(abs(z-yeuler))
```

Basic Problem

Numerically "solve" $y' = f(t, y)$ on the interval $[a, b]$ with $y(a) = y_0$.

- Find a discrete set of points

  $a = t_0 < t_1 < t_2 < \cdots < t_{N-1} < t_N = b$

- and values

  $y_0, y_1, y_2, \ldots, y_{N-1}, y_N$

  with y_j approximately equal to $y(t_j)$.

Return

Runge-Kutta vs Euler

- Both use a fixed step size $h = (b - a)/N$.
- Euler's method

  $$y_k = y_{k-1} + f(t_{k-1}, y_{k-1})h$$

- Runge-Kutta methods

  $$y_k = y_{k-1} + Sh$$

  - S is a weighted average of two or more slopes.
  - Slopes chosen to increase the accuracy.

Return

Second Order Runge-Kutta – First Step

- Use $S = \frac{1}{2}(s_1 + s_2)$, where
  - $s_1 = y'(t_0) = f(t_0, y_0)$
  - $s_2 = f(t_0 + h, y_0 + s_1 h)$
- $y_1 = y_0 + \frac{1}{2}(s_1 + s_2)\, h; \quad t_1 = t_0 + h$

Return

Second Order Runge-Kutta – Algorithm

Input t_0 and y_0.

for $k = 1$ to N

$$s_1 = f(t_{k-1}, y_{k-1})$$

$$s_2 = f(t_{k-1} + h, y_{k-1} + s_1 h)$$

$$y_k = y_{k-1} + \tfrac{1}{2}(s_1 + s_2)\, h$$

$$t_k = t_{k-1} + h$$

2^{nd} Order R-K – Error Analysis

- Slopes s_1 and s_2 chosen so that the truncation error at each step is $\leq Ch^3$.
- There are $N = (b-a)/h$ steps.
- Truncation error can propagate exponentially.

$$\text{Max error} \leq C\left(e^{L(b-a)} - 1\right)\, h^2,$$

where $C \,\&\, L$ are constants that depend on f.

Return

2^{nd} Order R-K – Error Analysis

$$\text{Max error} \leq C\left(e^{L(b-a)} - 1\right) h^2,$$

where $C \,\&\, L$ are constants that depend on f.

- Good news: decreases like h^2 as h decreases.
- Bad news: can get exponentially large as b-a increases.

Fourth Order Runge-Kutta – First Step

- Use $S = \frac{1}{6}(s_1 + 2s_2 + 2s_3 + s_4)$, where
  - $s_1 = f(t_{k-1}, y_{k-1})$
  - $s_2 = f(t_{k-1} + h/2, y_{k-1} + s_1 h/2)$
  - $s_3 = f(t_{k-1} + h/2, y_{k-1} + s_2 h/2)$
  - $s_4 = f(t_{k-1} + h, y_{k-1} + s_3 h)$
- $y_k = y_{k-1} + \frac{1}{6}(s_1 + 2s_2 + 2s_3 + s_4)\, h$

Fourth Order Runge-Kutta – Algorithm

Input t_0 and y_0.

for $k = 1$ to N

$$s_1 = f(t_{k-1}, y_{k-1})$$
$$s_2 = f(t_{k-1} + h/2, y_{k-1} + s_1 h/2)$$
$$s_3 = f(t_{k-1} + h/2, y_{k-1} + s_2 h/2)$$
$$s_4 = f(t_{k-1} + h, y_{k-1} + s_3 h)$$
$$y_k = y_{k-1} + \tfrac{1}{6}(s_1 + 2s_2 + 2s_3 + s_4)\, h$$
$$t_k = t_{k-1} + h$$

4^{th} Order R-K – Error Analysis

- Slopes are chosen so that the truncation error at each step is $\leq Ch^5$.
- There are $N = (b-a)/h$ steps.
- Truncation error can propagate exponentially.

$$\text{Max error} \leq C\left(e^{L(b-a)} - 1\right) h^4,$$

where C & L are constants that depend on f.

4^{th} Order R-K – Error Analysis

$$\text{Max error} \leq C \left(e^{L(b-a)} - 1 \right) h^4,$$

- Good news: decreases like h^4 as h decreases.
- Bad news: can get exponentially large as b-a increases.

MATLAB Routines `rk2.m` & `rk4.m`

- Syntax

  `[t,y] = rk2(derfile,`$[t_0, t_f]$`,`y_0`,`h`);`

  - `derfile` - derivative m-file defining the equation.
  - t_0 - initial time; t_f - final time.
  - y_0 - initial value.
  - h - step size.

Experimental Error Analysis

- IVP $y' = \cos(t)/(2y-2)$ with $y(0) = 3$
- Exact solution: $y(t) = 1 + \sqrt{4 + \sin t}$.
- Solve using Runge-Kutta methods and compare with the exact solution.
- Do this for several step sizes.

Return

Experimental Error Analysis

- Solve IVP using Euler's method and the Runge-Kutta methods
- Compare the errors with the 3 methods.
- Do this for several step sizes.

Euler's Method – Algorithm

Input t_0 and y_0.

for $k = 1$ to N

$$y_k = y_{k-1} + f(t_{k-1}, y_{k-1})h$$

$$t_k = t_{k-1} + h$$

Return