

# Math 211

Lecture #13

Runge-Kutta Methods

September 25, 2002

## 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

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

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
  - ♦  $y_k = y_{k-1} + f(t_{k-1}, y_{k-1}) \cdot h$
- Runge-Kutta methods
  - ♦  $y_k = y_{k-1} + S \cdot h$ 
    - ▶  $S$  is a weighted average of two or more slopes.
    - ▶ Slopes chosen to increase the accuracy.

Return

## Second Order Runge-Kutta

- Use  $S = \frac{1}{2}(s_1 + s_2)$ , where
  - ♦  $s_1 = f(t_{k-1}, y_{k-1})$
  - ♦  $s_2 = f(t_{k-1} + h, y_{k-1} + s_1 \cdot h)$
- $y_k = y_{k-1} + \frac{1}{2}(s_1 + s_2) \cdot h$ ;  $t_k = t_{k-1} + h$

Return

## Second Order Runge-Kutta – Algorithm

Input  $t_0$  and  $y_0$ .

for  $k = 1$  to  $N$

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

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

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

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

Return

RK2 step

General idea

Euler

## 2<sup>nd</sup> Order R-K – Error Analysis

- The truncation error at each step is  $\leq Ch^3$ .
- There are  $N = (b - a)/h$  steps, but truncation error can propagate exponentially.
- Computation shows that

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

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.

Return

Euler

### Fourth Order Runge-Kutta

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

Return

RK2

### Fourth Order Runge-Kutta – Algorithm

Input  $t_0$  and  $y_0$ .

for  $k = 1$  to  $N$

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

$$s_2 = f(t_{k-1} + h/2, y_{k-1} + s_1 \cdot h/2)$$

$$s_3 = f(t_{k-1} + h/2, y_{k-1} + s_2 \cdot h/2)$$

$$s_4 = f(t_{k-1} + h, y_{k-1} + s_3 \cdot h)$$

$$y_k = y_{k-1} + \frac{1}{6}(s_1 + 2s_2 + 2s_3 + s_4) \cdot h$$

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

Return

RK4step

RK2

Euler

### 4<sup>th</sup> Order R-K – Error Analysis

- The truncation error at each step is  $\leq Ch^5$ .
- There are  $N = (b - a)/h$  steps, but the truncation error can propagate exponentially.
- Computation shows that

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

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

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

Return

RK2

Euler

### MATLAB Routines rk2.m & rk4.m

- Syntax: `[t,y] = rk2(derfile,[t0,tf],y0,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:  $y(t) = 1 + \sqrt{4 + \sin t}$ .
- For several step sizes solve using Runge-Kutta methods and compare with the exact solution.
- For several step sizes solve IVP using Euler's method and the Runge-Kutta methods and compare the errors with the 3 methods.
  - ♦ Use `odesolvedemo.m`.

Return

### 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

## Error Analysis – Euler's method

- Truncation error at each step is  $\leq Ch^2$ .
- There are  $N = (b - a)/h$  steps, but truncation error can grow exponentially.
- Computation shows that

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

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

- Good news: the error decreases as  $h$  decreases.
- Bad news: the error can get exponentially large as the length of the interval [i.e.,  $b-a$ ] increases.

Return