1

Math 211

Lecture #9

September 26, 2000

2

Runge-Kutta Methods

$y' = f(t, y)$ with $y(t_0) = y_0$

- Second order Runge-Kutta, first step:
- Fixed step size $h = (b - a)/N$.
- Approximate the solution curve by a line with a slope that is computed as follows:
  - $s_1 = y'(t_0) = f(t_0, y_0)$
  - $s_2 = f(t_0 + h, y_0 + s_1 h)$
  - $y_1 = y_0 + \frac{1}{2}(s_1 + s_2)\, h; \quad t_1 = t_0 + h$

3

2^{nd} order R-K: $y' = f(t, y)$ with $y(t_0) = y_0$

Algorithm

Input t_0 and y_0.

for $k = 1$ to N

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

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

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

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

4

Error Analysis, 2^{nd} order R-K

- The truncation error at each step is $\leq Ch^3$.
- There are $N = (b-a)/h$ steps.
- $$\text{Max error} \leq C\left(e^{L(b-a)} - 1\right) h^2,$$
  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.

5

4^{th} order R-K: $y' = f(t,y) \quad \text{with} \quad y(t_0) = y_0$

Algorithm

Input t_0 and y_0.

for $k = 1$ to N

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

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

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

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

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

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

6

Error Analysis, 4^{th} order R-K

- The truncation error at each step is $\leq Ch^5$.
- There are $N = (b-a)/h$ steps.
- $$\text{Max error} \leq C\left(e^{L(b-a)} - 1\right) h^4,$$
  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.

7

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.

8

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.

9

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.

10

ode45

- The first choice of MATLAB's solvers.
- Variable step method.
  - ◇ Specify error tolerance instead of step size.
  - ◇ MATLAB chooses the step size at each step to achieve the limit on the error.
  - ◇ The default tolerance is good enough for this course.
- Syntax
  `[t,y] = ode45(derfile,`$[t_0, t_f]$`,`y_0`);`

11

Solving Systems

- Example:

$$x' = v$$
$$v' = -9.8 - 0.04v|v|$$

- Change to vector notation. (Use MATLAB vector notation)
  - ◇ `u(1)` $= x$
  - ◇ `u(2)` $= v$

return

12

Derivative m-file `ball.m`

```
function upr = ball(t,u)

x = u(1);
v = u(2);
xpr = v;
vpr = -9.8 - 0.04*v*abs(v);
upr = [xpr; vpr];
```

back

13

Derivative m-file `ballshort.m`

```
function upr = ballshort(t,u)

upr = zeros(2,1);
upr(1) = u(2);
upr(2) = -9.8 - 0.04*u(2)*abs(u(2));
```

back

14

Solving Higher Order Equations

- Reduce to a first order system and solve the system.
- Example: The motion of a pendulum is modeled by

$$\theta'' = -\frac{g}{L}\sin\theta - D\theta'.$$

- Introduce $\omega = \theta'$. Notice

$$\omega' = -\frac{g}{L}\sin\theta - D\theta'.$$

return

15

Equivalent First Order System

$$\theta' = \omega$$
$$\omega' = -\frac{g}{L}\sin\theta - D\omega$$

- Change to vector notation. (Use MATLAB vector notation)
  - ◇ `u(1)` $= \theta$
  - ◇ `u(2)` $= \omega$

back

return

16

Derivative m-file pend.m

```
function upr = pend(t,u)

L= 1;
global D
th = u(1);
om = u(2);
thpr = om;
ompr = -(9.8/L)*sin(th) - D*om;
upr = [thpr; ompr];
```

back