Differential Equations

Math 341 Fall 2008 — MWF 2:30-3:25pm Fowler 307

 `http://faculty.oxy.edu/ron/math/341/08/`

Worksheet 31: **Monday November 24**

TITLE Numerical Methods for Solving ODEs
CURRENT READING Blanchard, 7.1, 7.2, 7.3

EXTRA CREDIT OPPORTUNITY
Homework Assignments due Monday December 1
Section 7.1: 3, 7, 10.
Section 7.2: 1, 7, 8, 13.
Section 7.3: 1, 5, 6.

SUMMARY
We shall look at how differential equations are really solved numerically, in practice. Euler's Method is **NOT** the way to do it!

---

1. Recalling Euler's Method

Recall that we are looking at numerical techniques to approximate the solution to $y' = f(x, y)$ with $y(x_0) = y_0$.

$$y(x_{new}) = y(x_{old}) + \Delta y \text{ where } \Delta y \approx y'(x_{old})\Delta x$$

In other words
$y_{new} = y_{old} + y'_{old}\Delta x$ and $x_{new} = x_{old} + \Delta x$.

Generally, the Euler Algorithm is written as
$y_{k+1} = y_k + f(x_k, y_k)\Delta x$ where $x_{k+1} = x_k + \Delta x$.

2. Error of Euler Method

The error at the n^{th} step of Euler's Method, i.e $|y(x_n) - y_n| = e_n \approx C\Delta x$. This means as $\Delta x \to 0$, the Euler error $e_n \to 0$.

3. Improving Euler's Method

There's an improvement to Euler's Method called **Heun's Method**. This is the first example of a predictor-corrector method.

The predictor step is identical to Euler's Method

$$u_{k+1} = y_k + f(x_k, y_k)\Delta x$$

the corrector step is

$$y_{k+1} = y_k + \frac{1}{2}[f(x_k, y_k) + f(x_{k+1}, u_{k+1})]\Delta x$$

It turns out that Heun's Method, really can be thought of as "Euler-like" because it has the form $y_{k+1} = y_k + M_k\Delta x$, where in Euler's Method $M_k = f(x_k, y_k)$ is the slope at this point. In the improved Euler's Method (a.k.a. Heun's Method) $M_k = \dfrac{f(x_k, y_k) + f(x_k, y_k + f(x_k, y_k)\Delta x)}{2}$ (which is the average slope over the interval $x_k \leq x \leq x_{k+1}$)

4. Runge-Kutta Method

The most commonly used numerical technique for solving ODEs of the form $y' = f(x, y)$ involves using four function values to better approximate M_k

The **Runge-Kutta Method** looks like $y_{k+1} = y_k + M_k \Delta x$ where $M_k = \dfrac{a_k + 2b_k + 2c_k + d_k}{6}$ where

$$\begin{aligned}
a_k &= f(x_k, y_k) \\
u_k &= y_k + a_k \frac{\Delta x}{2} \\
b_k &= f(x_k + 0.5\Delta x, u_k) \\
v_k &= y_k + b_k \frac{\Delta x}{2} \\
c_k &= f(x_k + 0.5\Delta x, v_k) \\
w_k &= y_k + c_k \Delta x \\
d_k &= f(x_{k+1}, w_k)
\end{aligned}$$

This is basically just a fancy "average" of the slope over the interval $[x_k, x_{k+1}]$ where the slopes at the mid-points (b_k and c_k) are weighted more heavily.

5. Using Matlab to Solve ODEs

Genrally to use Matlab to solve an ODE one needs a Matlab file (named "f.m") with the function $f(x, y)$ in it, which one inputs into the following code.

contents of f.m

```
function out=f(t,y)
out=exp(-y)+t^2;
```

contents of heun.m

```
function [t,y]=heun(f,a,b,ya,M)
%Input   - f is the function entered as a string 'f'
%        - a and b are the left and right endpoints
%        - ya is the initial condition y(a)
%        - M is the number of steps
%Output  - t is vector of input values y is vector of solution values
h=(b-a)/M;
T=zeros(1,M+1);
Y=zeros(1,M+1);
T=a:h:b;
Y(1)=ya;
for j=1:M
   k1=feval(f,T(j),Y(j));
   k2=feval(f,T(j+1),Y(j)+h*k1);
   Y(j+1)=Y(j)+(h/2)*(k1+k2);
end
t=T';y=Y';
```

To solve the ODE $y' = e^{-y} + t^2, y(0) = 1$ on the interval $0 \le t \le 2$ the command would be `[t,y]=heun('f',0,2,1,100);` The command to plot the solution (in blue circles) is `plot(t,y,'bo')`