**Section 8.3 - The Runge-Kutta Method**

In Problem Set C #16 I have asked you to implement the Runge-Kutta method in MATLAB. Here I use my implementation, which I called **rungekutta**, to find a numerical solution of

$$y' = 2y -1, \quad y(0) = 1.$$

**rungekutta** takes the same input as **myeuler**.

f = @(t,y) 2\*y - 1;

First we use step size 0.1.

[t1,y1] = rungekutta(f, 0, 1, 0.4, 4); [t1 y1]

ans =

0 1.0000

0.1000 1.1107

0.2000 1.2459

0.3000 1.4111

0.4000 1.6128

Next we use step size 0.05.

[t2,y2] = rungekutta(f, 0, 1, 0.4, 8); [t2 y2]

ans =

0 1.0000

0.0500 1.0526

0.1000 1.1107

0.1500 1.1749

0.2000 1.2459

0.2500 1.3244

0.3000 1.4111

0.3500 1.5069

0.4000 1.6128

We will make a table of values at t = 0, 0.1,...,0.4. We will include the exact solution, which is

syms t

y = dsolve('Dy = 2\*y - 1, y(0) = 1', 't')

y =

exp(2\*t)/2 + 1/2

We will also include the approximate solution we had obtained with Euler's method using step size 0.025.

[ta,ya] = myeuler(f, 0, 1, 0.4, 16); ya

ya =

1.0000

1.0250

1.0513

1.0788

1.1078

1.1381

1.1700

1.2036

1.2387

1.2757

1.3144

1.3552

1.3979

1.4428

1.4900

1.5395

1.5914

Here is the table. The first column is t, the second ya, the third y1, the fourth y2 and the last y.

T = [ ]

for k = 0:4

digits(9)

A = [vpa(t1(k+1)),vpa(ya(4\*k+1)),vpa(y1(k+1)),vpa(y2(2\*k+1)),vpa(subs(y,'t',t1(k+1)))];

T = [T;A];

end

T

T =

[]

T =

[ 0, 1.0, 1.0, 1.0, 1.0]

[ 0.1, 1.10775312, 1.1107, 1.11070129, 1.11070138]

[ 0.2, 1.23872772, 1.24590898, 1.24591212, 1.24591235]

[ 0.3, 1.39792816, 1.41105323, 1.41105898, 1.4110594]

[ 0.4, 1.59143729, 1.61276041, 1.61276978, 1.61277046]

We see that with step size 0.1 the Runge-Kutta method gives errors that are less than 1.01\*10^(-5). With step size 0.05 it gives errors that are less than 10^(-6). With step size 0.025, Euler's method gives an error greater than 0.02 at t=0.4.

Published with MATLAB® 7.13