

M439 Numerical Analysis Lab: Runge Kutta Order Four Method for Numerical Solution of First Order Ordinary Differential Equations

m-files for this lab:

[euler.m](#)

[heun.m](#)

[rk4.m](#)

Runge Kutta Method of Order 4

2. Consider the initial value problem number 3 (using t as the independent variable, y as the dependent variable) from section 9.3. Create an m-file `f3.m` function containing the definition for the right hand side of this differential equation – i.e.

```
function z = f(t,y)
    z = exp(-2*t) - 2*y;
```

4. For comparison, run the calls to Euler and Heun from the last lab:

```
H1 = heun(@f3,0,1,1,5)
H2 = heun(@f3,0,1,1,10)
```

```
E1 = euler(@f3,0,1,1/10,5)
E2 = euler(@f3,0,1,1/10,10)
```

3. Run the Fourth Order Runge Kutta on the same subintervals, checking to see if answer agrees with our by hand computations.

```
R1 = rk4(@f3,0,1,1/10,5)
R2 =rk4(@f3,0,1,1/10,10)
```

4. Plot the exact solution and your approximations with 20 subintervals on the same graph for comparison for Euler, Heun, and Runge Kutta:

```
ezplot('1*exp(-2*t)+t*exp(-2*t)',0,2)
hold on
plot(E2(:,1),E2(:,2),'g-')
plot(H2(:,1),H2(:,2),'r-')
plot(R2(:,1),R2(:,2),'b-')
```

5. Construct the following tables to compare your results:

```
t = (linspace(0,2,11))'  
F = .1*exp(-2.*t)+t.*exp(-2.*t)  
ErrorEst1 = [(E1(:,2)-F) (H1(:,2)-F) (R1(:,2)-F)]  
t = (linspace(0,2,21))'  
F = .1*exp(-2.*t)+t.*exp(-2.*t)
```

```
ErrorEst2 = [(E2(:,2)-F) (H2(:,2)-F) (R2(:,2)-F)]
```

Note the differences in the final global error in the end at $t = 2$ for the three different methods, which is consistent with the orders of each of the methods.

6. MATLAB has a number of powerful built-in functions for numerically approximating solutions to differential equations. One is **ode45** which implements a variable step Runge-Kutta “4-5” method.

Execute the following commands to call this method for the same differential equation as above and compare the graph of the solution to the true solution:

```
>> [t,y]=ode45(@f3,[0,2],1/10);  
>> clf  
>> plot(t,y,'-')  
>> hold on  
>> ezplot('1*exp(-2*t)+t*exp(-2*t)',0,2)
```

7. The **ode45** function can also be used to handle several initial conditions for the same differential equation at once. Use the following commands to plot

a) Create (and save) an m-file function mfile called “flab.m” that contains the following:

```
function z = f(t,y)  
z = -2*y+2*cos(t).*sin(2*t);
```

b) Now execute the following:

```
>> [t,y]=ode45(@flab,[0,2*pi],-5:1:5)  
>> clf  
>> plot(t,y)
```

Here we are solving the differential equation with each of the following initial conditions $y(0)=-5, -4, -3, -1, -1, 0, 1, 2, 3, 4, 5$

We can see that no matter what the initial conditions all converge to the same “steady state” .

8. If you call ode45 in this form (with out any output variables) you get a plot directly:

```
>> ode45(@flab,[0,2*pi],-5:1:5)
```