CHBE 553 – Assignment 7 – 03/11/2019

Objective: To Solve,

$f' = (f/2)*(g-8)$,

$g' = f^2/2$,

Given, f (0) = 8, g(0) = 0, between t = [0,10]

Runge- Kutta 4th order method:

Part a) Fixed step size, h:

$$\mathbf{K}_1 = h\mathbf{F}(x, \mathbf{y})$$

$$\mathbf{K}_2 = h\mathbf{F}\left(x + \frac{h}{2}, \mathbf{y} + \frac{\mathbf{K}_1}{2}\right)$$

$$\mathbf{K}_3 = h\mathbf{F}\left(x + \frac{h}{2}, y + \frac{\mathbf{K}_2}{2}\right)$$

$$\mathbf{K}_4 = h\mathbf{F}(x + h, \mathbf{y} + \mathbf{K}_3)$$

$$\mathbf{y}(x + h) = \mathbf{y}(x) + \frac{1}{6}(\mathbf{K}_1 + 2\mathbf{K}_2 + 2\mathbf{K}_3 + \mathbf{K}_4)$$

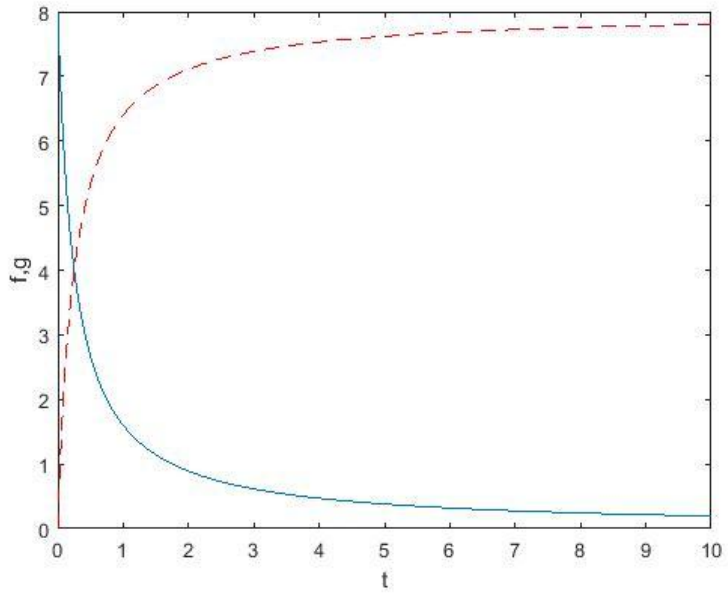
To solve this couples ODE's, h is chosen as 0.001. It can be noted that when h = 1, the program seizes to converge to the solution.

Solution obtained using h = 0.001

f(10) = 0.1951

g (10)= 7.8049

Plot obtained:



Part b) Adaptive step size, h:

To improve the range the step size, and take in account the truncation error at each step, adaptive step sizing method is implemented. One method of adaptive step sizing is halving method. After each iteration, the step size is changed by $h = h * (\frac{\epsilon}{\Delta})^{\frac{1}{5}}$.

Since in this problem, there can be two error approximations between the i^{th} and $i\text{-}1^{th}$ calculated f and g values, Δ is assumed to be the error approximation with maximum magnitude.

The following results were obtained for a tolerance limit of 10^{-14} and **h = 0.001**

f(10) = 0.1946

g (10)= 7.8054

The following results were obtained for a tolerance limit of 10^{-14} and **h = 1**

f (10)= 0.1943

g(10) = 7.8057

MATLAB CODE

Runge kutta function

```
%runge kutta 4th order method
function sol = RK4(f,g,t_0,h)
%fval function to evaluate the first ODE
%gval function to evaluate the second ODE
t = t_0+(h/2);
k1 = h*fval(f,g);
l1 = h*gval(f,g);

k2 = h*fval(f+(k1/2),g+(l1/2));
l2 = h*gval(f+(k1/2),g+(l1/2));

k3 = h*fval(f+(k2/2),g+(l2/2));
l3 = h*gval(f+(k2/2),g+(l2/2));

t4 = t_0+h;
k4 = h*fval(f+k3,g+l3);
l4 = h*gval(f+k3,g+l3);
%final values of f and g
f_final = f+((1/6)*(k1+2*k2+2*k3+k4));
g_final = g+((1/6)*(l1+2*l2+2*l3+l4));
sol = [t4,f_final,g_final];
end


%function to evaluate second ODE
function g_dash = gval(f,g)
g_dash = f^2/2;
end

%function to evaluate first ODE
function f_dash = fval(f,g)
f_dash = (f/2)*(g-8);
end
```

Part a) Fixed step size

```
%assignment7-part1-at fixed step size, h = 0.1
%initial values
f = 8;
g = 0;
%fixed step size h = 0.001
h = 0.001;
%t ranges between [0,10]
t_0 = 0;
i=0.001;
t_stop = 10;
%RK loop
N = 10000;
    for i = 1:N

    %function call to Runge kutta function, that return t, f and g values
    sol= RK4(f,g,t_0,h);
    t_new(i) = sol(1);
    f_new(i) = sol(2);
    g_new(i) = sol(3);

    t_0 = t_new(i);
    f = f_new(i);
    g = g_new(i);
    if t_0>t_stop
        break;
    end

end
plot(t_new,f_new)
xlabel('t');
ylabel('f,g');
hold on
plot(t_new,g_new,'r--')
hold off
```

Part b) Adaptive step size

```
%Assignemnt 7 -Adaptive step size
%initial values
f = 8;
g = 0;
%initial step size h = 0.001
h = 1;
i=1;
%t ranges between [0,10]
t_0 = 0;t_stop = 10;
%tolerance limit
tol = 10^-14;

%RK loop
while t_0<t_stop
```

```
    %function call to evaluate f and g using RK4 at h
step0= RK4(f,g,t_0,h);
%function call to evaluate f and g using RK4 at h/2
step1 = RK4(f,g,t_0,h/2);
t_new = step1(1);
f_new = step1(2);
g_new = step1(3);
%function call to evaluate f and g using RK4 at h/2 and f_new,g_new
step2 = RK4(f_new,g_new,t_new,h/2);
%error approximations
err(1) = abs(step2(2)-step0(2));
err(2) = abs(step2(3)-step0(3));
%maximum of error is ued for further steps
error(i) = max(err);

if (error(i)<tol)
     h = h*(tol/error(i))^(1/5);
     t(i) = step2(1);
     f_i(i) = step2(2);
     g_i(i)= step2(3);
     t_0 = t(i);
     f = f_i(i);
     g= g_i(i);
else
    h = h*(tol/error(i))^(1/5);
end

i = i+1;
end

plot(t,f_i)
xlabel('t');
ylabel('f,g')
hold on
plot(t,g_i,'r--')
hold off
```