ME 594 – Numerical Methods – HW08

Viral Panchal | Due Date: 11/16

"I pledge my honor that I have abided by the Stevens Honor system"

1*. Solve the ODE $y' = -2xy$ with $y(0) = 1$ using both the explicit and implicit Euler methods. Use step sizes $h = 0.4$, 0.2, 0.1, and 0.05 on the domain $[0, 6]$. Plot the results for all step sizes on separate plots for the implicit and explicit methods. The exact solution is $y(x) = e^{-x^2}$ which should also be included in the plots. There should be two plots submitted, each with five curves on it. Include the code used to solve the problem. You can use a shortcut for the implicit equation here; it is not necessary to use a nonlinear equation solver as part of the code.

Solution:

Q.1) Given:

$y' = -2xy \qquad y(0) = 1$

$h = 0.4, 0.2, 0.1, 0.05$

domain $[0, 6]$

a) Using Explicit Euler

$$y_{k+1} = y_k + h f(x_k, y_k)$$

$$= y_k + h[-2x_k y_k]$$

$$y_{k+1} = (1 - 2x_k h)\, y_k$$ ..... Solved in Matlab

b) Using Implicit Euler

$$y_{k+1} = y_k + h f(x_{k+1}, y_{k+1})$$

$$= y_k + h[-2x_{k+1}, y_{k+1}]$$

$$\therefore y_{k+1}(1 + 2x_{k+1} h) = y_k$$

$$\therefore y_{k+1} = \frac{y_k}{(1 + 2x_{k+1} h)}$$ ... Matlab program attached.

Matlab program:

- **Explicit Euler function**

```
% Explicit Euler function

function f = Explicit_euler(x_k,y_k,h)
f = (1-2*x_k*h)*y_k;
end
```

- **Implicit Euler function**

```
% Implicit Euler function

function f = Implicit_euler(x_k_p1,y_k,h)
f = y_k/(1+2*x_k_p1*h);
end
```

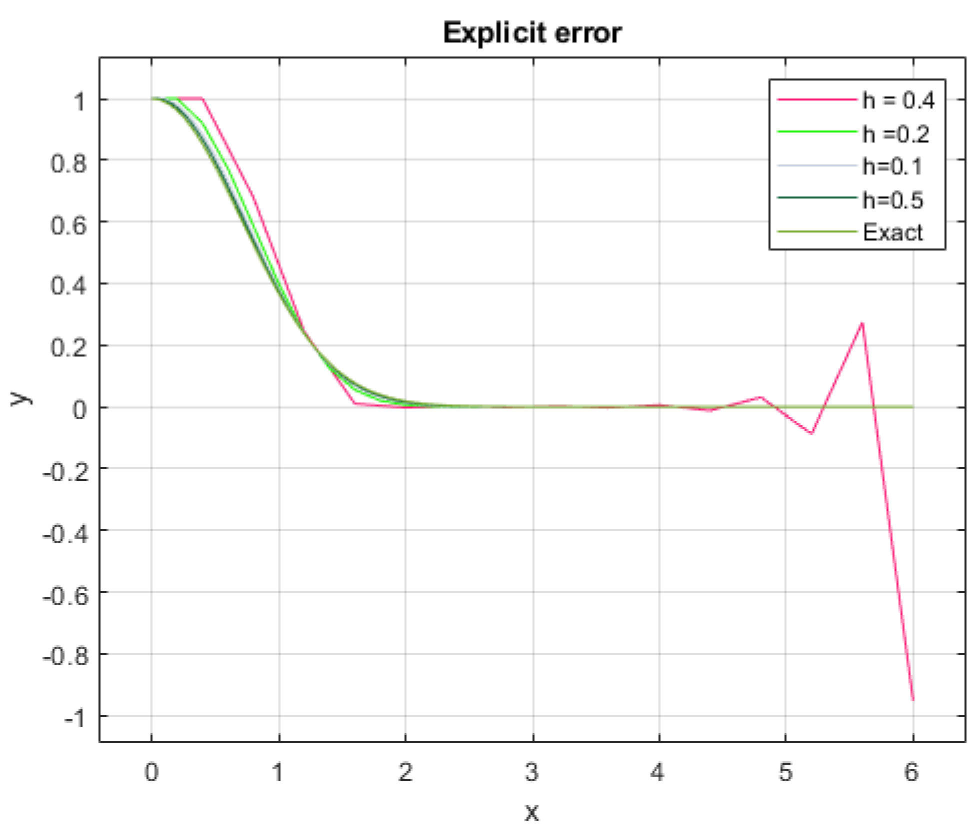
- **Driver for Q1**

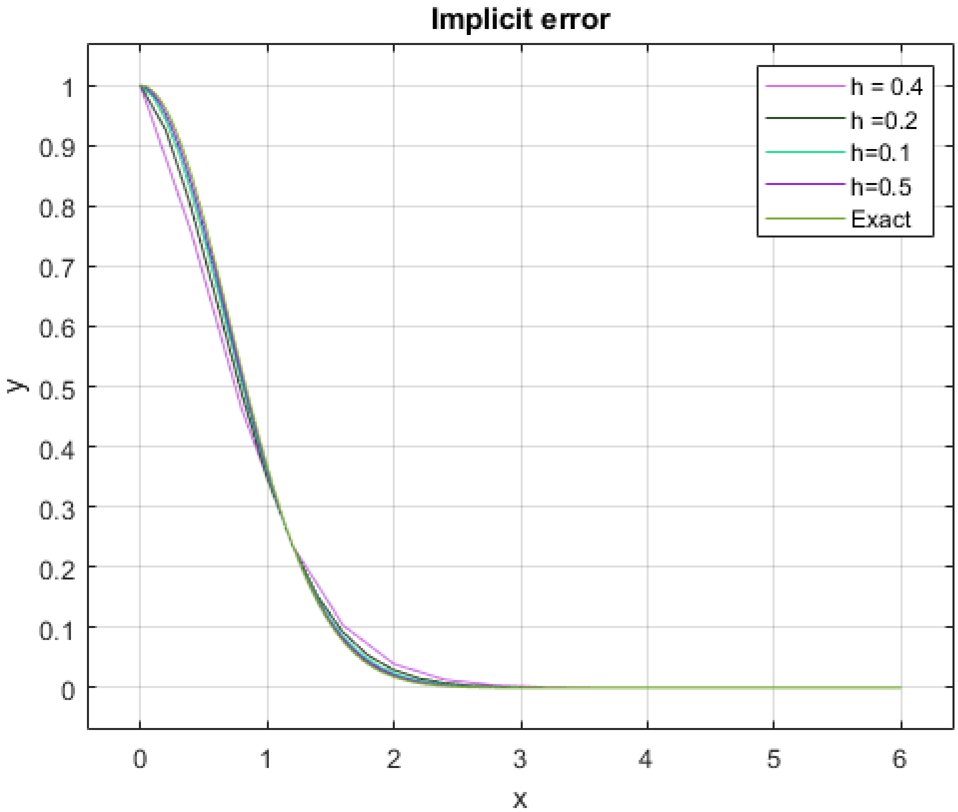
```
% Q1 driver
% This driver runs both Explicit and Implcit Euler method outputs plot for
% both individually.
clear all
close all
clc
% for Explicit Euler method

x_0 = 0;
y_0 = 1;
x_n = 6;

for i = 0:3
    h = 0.4/2^i;
    n=(x_n-x_0)/h;
    x = zeros(1,n+1);
    y = zeros(1,n+1);

    x = x_0:h:x_n;
    y(1) = y_0;
    for k = 1:n
        y(k+1) = feval('Explicit_euler',x(k),y(k),h);
    end

    plot(x,y,'color',rand(1,3))
```

```
    hold on

end

y_exact = zeros(1,n+1);
y_exact(1) = y_0;
for k = 2:n+1
    y_exact(k) = exp(-x(k).^2);
end

plot(x,y_exact)
title('Explicit error')
axis padded
grid on
xlabel('x')
ylabel('y')
legend('h = 0.4','h =0.2','h=0.1','h=0.5','Exact')


figure

% For Implicit Euler method
% Redefing the initial variables to override the magnitudes of certain
% variables from the previous metthod.
x_0 = 0;
y_0 = 1;
x_n = 6;

for i = 0:3
    h = 0.4/2^i;
    n=(x_n-x_0)/h;
    x = zeros(1,n+1);
    y = zeros(1,n+1);

    x = x_0:h:x_n;
    y(1) = y_0;
    for k = 1:n
        y(k+1) = feval('Implicit_euler',x(k+1),y(k),h);
    end

    plot(x,y,'color',rand(1,3))
```

```
    hold on

end

y_exact = zeros(1,n+1);
y_exact(1) = y_0;
for k = 2:n+1
    y_exact(k) = exp(-x(k).^2);
end

plot(x,y_exact)
title('Implicit error')
axis padded
grid on
xlabel('x')
ylabel('y')
legend('h = 0.4','h =0.2','h=0.1','h=0.5','Exact')
```

Matlab output:





Published with MATLAB® R2021a

2*. Solve the ODE $y' = -xy^3$ with $y(0) = 1$ using the implicit Euler method. For this problem, you must include a nonlinear equation solver. Include your code, and use step sizes $h = 1/2$, $1/4$, and $1/8$ on the interval $[0, 20]$. Plot the results for all step sizes along with the exact solution,

$$y(x) = \frac{1}{(1+x^2)^{1/2}}.$$

Solution:

Q2) Given:

$y' = -xy^3 \qquad y(0) = 1$

$h = \frac{1}{2}, \frac{1}{4}, \frac{1}{8} \qquad$ domain $[0, 20]$

Using Implicit Euler method:

$y_{k+1} = y_k + hf(x_{k+1}, y_{k+1})$

$= y_k + h[-x_{k+1}\, y_{k+1}^3]$

$y_{k+1} = y_k - h\, x_{k+1}\, y_{k+1}^3$ ← This non-linear equation can be solved using Newton's method.

$g(y_{k+1}) = h x_{k+1} y_{k+1}^3 - y_k = 0$

let's say $z = y_{k+1}$

$g(z) = h\, x_{k+1} z^3 + z - y_k$

$\therefore g'(z) = 3h\, x_{k+1}\, z^2 + 1$

$\therefore z^{(k+1)} = z^{(k)} - \frac{g(z)}{g'(z)}$

↓

Matlab code attached

Matlab program:

- ***Script for G as shown in manual work***

```
% G for newton solver

function f = G_NS(z,h,y_k,x_k_p1)
f = h*x_k_p1*z^3+z-y_k;
end
```

- ***Script for GZ as shown in the above work***

```
% GP for Newton solver

function f = GZ_NS(z,x_k_p1,h)
f = 3*x_k_p1*z^2+1;
end
```

- ***Newton function***

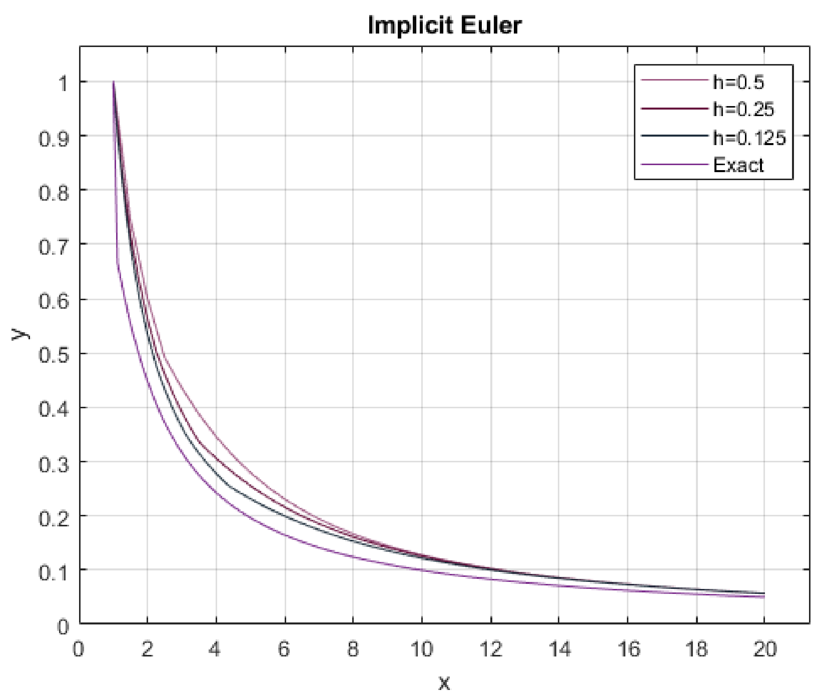
```
function z = Newton_z(h,y_k,x_k_p1)

tolerance = h^2/4;
z = y_k;
max_1 = 1000;

for j = 1:max_1
    p = z-feval('G_NS',z,h,y_k,x_k_p1)/feval('GZ_NS',z,x_k_p1,h);

    error = abs(p-z);
    rel_error = 2*error/(abs(p)+abs(z));
    f = feval('G_NS',p,h,y_k,x_k_p1);
    z = p;

    if(error<tolerance)||(rel_error<tolerance)||(abs(f)<tolerance)
        break
    end
end
```

- ***Driver for Q2***

```
% Q2 Driver
clear all
close all
clc

x_0 = 1;
y_0 = 1;
x_n = 20;

for i = 1:3
    h = 1/2^i;
    n = (x_n-x_0)/h;
```

```
    x = zeros(1,n+1);
    y = zeros(1,n+1);

    x = x_0:h:x_n;
    y(1) = y_0;
    for k = 1:n
        z = y(k);
        y(k+1) = Newton_z(h,z,x(k+1));
    end
    plot(x,y,'color',rand(1,3))
    hold on
end

y_exact = zeros(1,n+1);
y_exact(1) = y_0;
for k = 2:n+1
    y_exact(k) = (1+x(k).^2).^(-.5);
end

plot(x,y_exact)
title('Implicit Euler')
axis padded
grid on
xlabel('x')
ylabel('y')
legend('h=0.5','h=0.25','h=0.125','Exact')
```

Matlab output:



Published with MATLAB® R2021a

3*. Solve the ODE $y' = x - xy/2$ with $y(1) = 1$ using the trapezoid method and the improved Euler method. Include your code for each method, and use step sizes $h = 0.8$, 0.4, and 0.2 on the interval $[1, 5]$. Plot the results for all step sizes along with the exact solution,

$$y = 2 - e^{\frac{1-x^2}{4}}.$$

Solution:

Q.3) ~~Using~~ Given:

$$y' = x - \frac{xy}{2} \quad ; \; y(1) = 1$$

domain $[1, 5]$

a) Using Trapezoid method

$h = 0.8, 0.4, 0.2$

$$y_{k+1} = y_k + \frac{h}{2}\left[f(x_k, y_k) + f(x_{k+1}, y_{k+1})\right]$$

$$y_{k+1} = y_k + \frac{h}{2}\left[\left(x_k - \frac{x_k y_k}{2}\right) + \left(x_{k+1} - \frac{x_{k+1} y_{k+1}}{2}\right)\right]$$

$$= y_k + \frac{h}{2}\left[x_k + x_{k+1} - \frac{x_k y_k}{2}\right] - \frac{h}{4} x_{k+1} y_{k+1}$$

$$\left(1 + \frac{h}{4} y_{k+1}\right) y_{k+1} = y_k + \frac{h}{2}\left[x_k + x_{k+1} - \frac{x_k y_k}{2}\right]$$

$$y_{k+1} = \frac{y_k + h/2\left[x_k + x_{k+1} - x_k y_k/2\right]}{1 + \frac{h}{4} y_{k+1}}$$

b) Using Improve Euler method:

$$y^*_{k+1} = y_k + h f(x_k, y_k)$$

$$= y_k + h\left[x_k - \frac{x_k y_k}{2}\right]$$

$$y_{k+1} = y_k + h/2\left[f(x_k, y_k) + f(x_{k+1}, y^*_{k+1})\right]$$

$$y_{k+1} = y_k + \frac{h}{2}\left[\left(x_k - \frac{x_k y_k}{2}\right) + \left(x_{k+1} - \frac{x_{k+1} y^*_{k+1}}{2}\right)\right]$$

Matlab program attached

MATLAB program:

- ***Euler function script***

```
% Euler function

function f = Euler(x,y)
f = x-(x*y)/2;
end
```

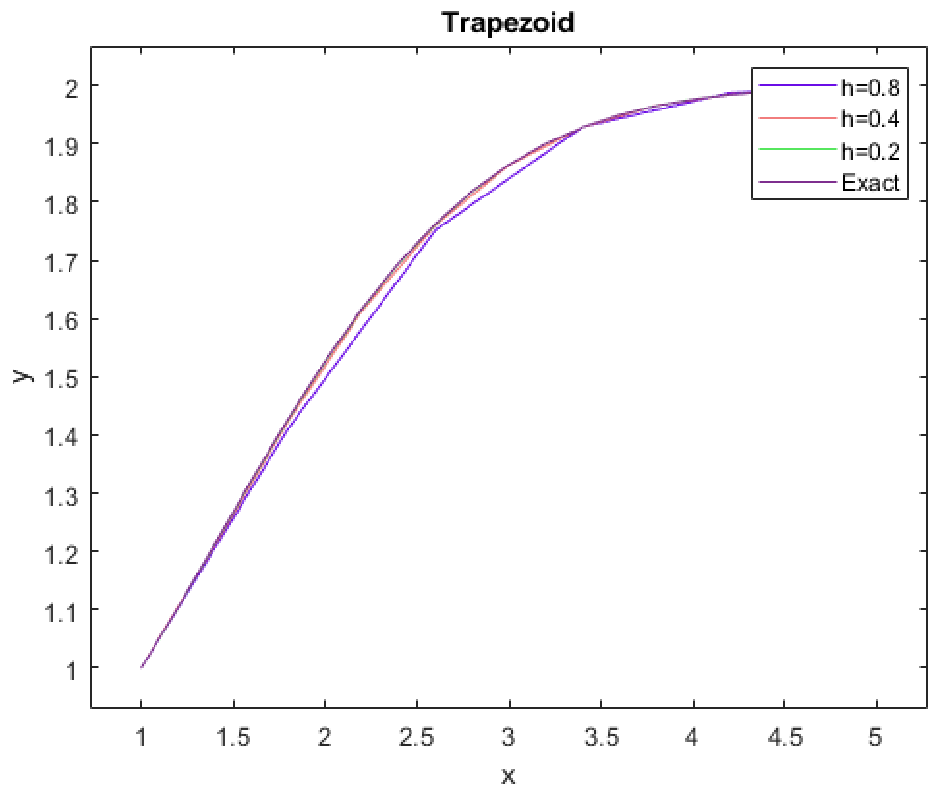
- ***Trapezoid method function script***

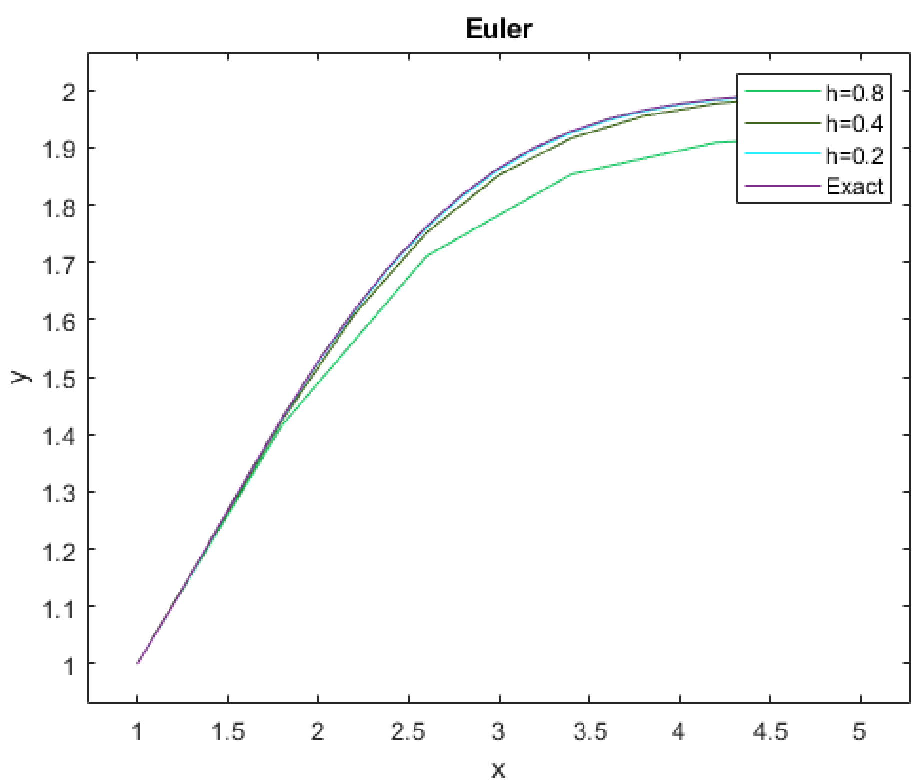
```
% Trapezoid function

function f = Trapezoid(x_k,x_k_p1,y_k,h)
f = (y_k+h/2*(x_k+x_k_p1-x_k*y_k/2))/(1+h/4*x_k_p1);
end
```

- ***Driver for Q3***

```
% Driver Q3
% This driver includes the program for both the Trapezoid and Euler method.
% The first output figure is for the trapezoid method followed by the
% modifiied euler method.
clear all
close all
clc

% For trapezoid method
x_0 = 1;
y_0 = 1;
x_n = 5;

for i = 0:2

    h = 0.8/2^i;
    n = round((x_n-x_0)/h);
    x = zeros(1,n+1);
    y = zeros(1,n+1);

    x = x_0:h:x_n;
    y(1) = y_0;
    for k = 1:n
        y(k+1) = feval('Trapezoid',x(k),x(k+1),y(k),h);
    end
    plot(x,y,'color',rand(1,3))
    hold on
end

y_exact = zeros(1,n+1);
y_exact(1) = y_0;
for k = 2:n+1
    y_exact(k) = 2-exp(((1-x(k).^2)/4));
```

```
end

plot(x,y_exact)
title('Trapezoid')
axis padded
xlabel('x')
ylabel('y')
legend('h=0.8','h=0.4','h=0.2','Exact')

figure


% For Euler method
% Need to redefine the initial values to overide magnitudes from previous
% method
x_0 = 1;
y_0 = 1;
x_n = 5;

for i = 0:2

    h = 0.8/2^i;
    n = round((x_n-x_0)/h);
    x = zeros(1,n+1);
    y = zeros(1,n+1);

    x = x_0:h:x_n;
    y(1) = y_0;
    for k = 1:n
        y_star = y(k) + h*(x(k) - x(k)*y(k)/2);
        y(k+1) = y(k) + 0.5 * h* (feval('Euler',x(k),y(k))+feval('Euler',x(k+1),y_star));
    end
    plot(x,y,'color',rand(1,3))
    hold on
end

y_exact = zeros(1,n+1);
y_exact(1) = y_0;
for k = 2:n+1
    y_exact(k) = 2-exp((1-x(k).^2)/4);
end

plot(x,y_exact)
title('Euler')
xlabel('x')
ylabel('y')
axis padded
legend('h=0.8','h=0.4','h=0.2','Exact');
```





Published with MATLAB® R2021a

4*. In a chemical reaction, one molecule of A combines with one molecule of B to form one molecule of the chemical C. It is found that the concentration $y(t)$ of C at time t is the solution to the I.V.P.

$$y' = k(a-y)(b-y) \quad \text{with} \quad y(0) = 0,$$

where k is a positive constant and a and b are the initial concentrations of A and B, respectively. Suppose that $k = 0.01$ liter/(millimole $\cdot$ s), $a = 70$ millimoles/liter, and $b = 50$ millimoles/liter. Include your code, and use the classical 4th order Runge-Kutta method with $h = 0.5$ to find the solution over $[0, 20]$.

You can compare your numerical solution with the exact solution:

$$y(t) = 350\frac{(1-e^{-0.2t})}{(7-5e^{-0.2t})}.$$

Observe that the limiting value is 50 as $t \to +\infty$.

Solution:

- ***Chemical concentration***

```
function f = chem(t,y)

k = 0.01;
a = 70;
b = 50;
f = k*(a-y)*(b-y);

end
```

- ***Runge-Kutta function***

```
% Runge-Kutta function
function [x,y] = RK4(ode,a,b,h,y_ini)


n = round((b-a)/h)+1;
y = zeros(1,n);

x = linspace(a,b,n);
y(1) = y_ini;

for i = 1:n-1
    k(1:4) = 0;
    k(1) = feval(ode,x(i),y(i));
```

```
    k(2) = feval(ode,x(i)+0.5*h,y(i)+0.5*k(1)*h);
    k(3) = feval(ode,x(i)+0.5*h,y(i)+0.5*k(2)*h);
    k(4) = feval(ode,x(i)+h,y(i)+k(3)*h);

    y(i+1) = y(i) + 1/6*(k(1)+2*k(2)+2*k(3)+k(4))*h;
end
```

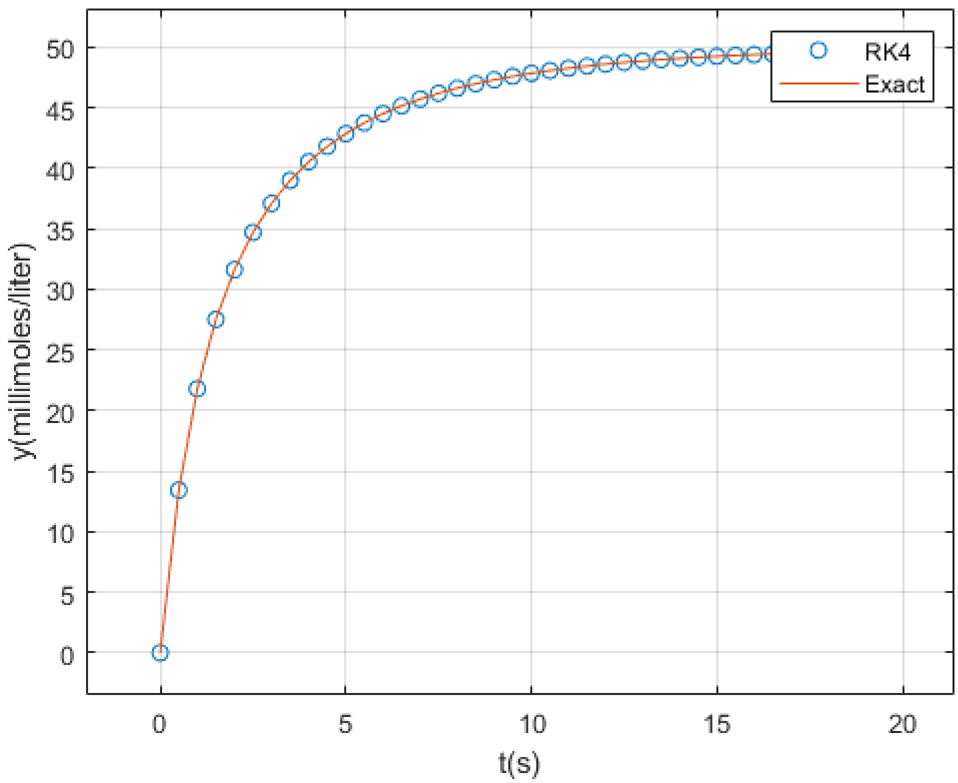
- ***Exact solution***

```
% Script for exact solution

function f = Exact(t)
f = 350*(1-exp(-0.2*t))/(7-5*exp(-0.2*t));
end
```

- ***Driver for Q4***

```
% Q4 driver
clear all
close all
clc

a = 0;
b = 20;
h = 0.5;
y_ini = 0;

[x,y] = RK4('chem',a,b,h,y_ini);

n = round((b-a)/h)+1;
x_2 = linspace(a,b,n);
for i = 1:n
    y_2(i) = feval('Exact',x_2(i));
end

plot(x,y,'o',x_2,y_2)
xlabel('t(s)')
ylabel('y(millimoles/liter)')
axis padded
grid on
legend('RK4','Exact');
```

Matlab output:



Published with MATLAB® R2021a