Numerical Analysis – Ex4

$\boxed{P1}$

a) $y'' - ty = 0 \Rightarrow y'' = ty$

$y_1 = y \qquad\qquad y_1' = y_2$

$y_2 = y' \Rightarrow \quad y_2' = ty_1$

b) $y'' - 2ty' + 2y = 0 \Rightarrow y'' = 2ty' - 2y$

$y_1 = y \qquad\qquad y_1' = y_2$

$y_2 = y' \quad \Rightarrow \quad y_2' = 2ty_2 - 2y_1$

c) $y'' - ty' - y = 0 \Rightarrow y'' = ty' + y$

$y_1 = y \qquad\qquad y_1' = y_2$

$y_2 = y' \quad \Rightarrow \quad y_2' = ty_2 + y_1$

P2

a) $y = \frac{1}{2}(e^{t} + e^{-t} - t^{2}) - 1$

$y' = \frac{1}{2}(e^{t} - e^{-t} - 2t)$

$y'' = \frac{1}{2}(e^{t} + e^{-t} - 2)$

$y''' = \frac{1}{2}(e^{t} - e^{-t})$

$\Rightarrow y''' - y' = t$

$y(0) = \frac{1}{2}(e^{0} + e^{0} - 0^{2}) - 1 = 0$

$y'(0) = \frac{1}{2}(e^{0} - e^{0} - 2 \times 0) = 0$

$y''(0) = \frac{1}{2}(e^{0} + e^{0} - 2) = 0$

$\Rightarrow$ y is the solution to initial value problem

$$\begin{cases} y''' - y' = t \\ y(0) = y'(0) = y''(0) \end{cases}$$

b) $y_1 = y$ $\qquad y_1' = y_2$

$y_2 = y' \quad \Rightarrow \quad y_2' = y_3$

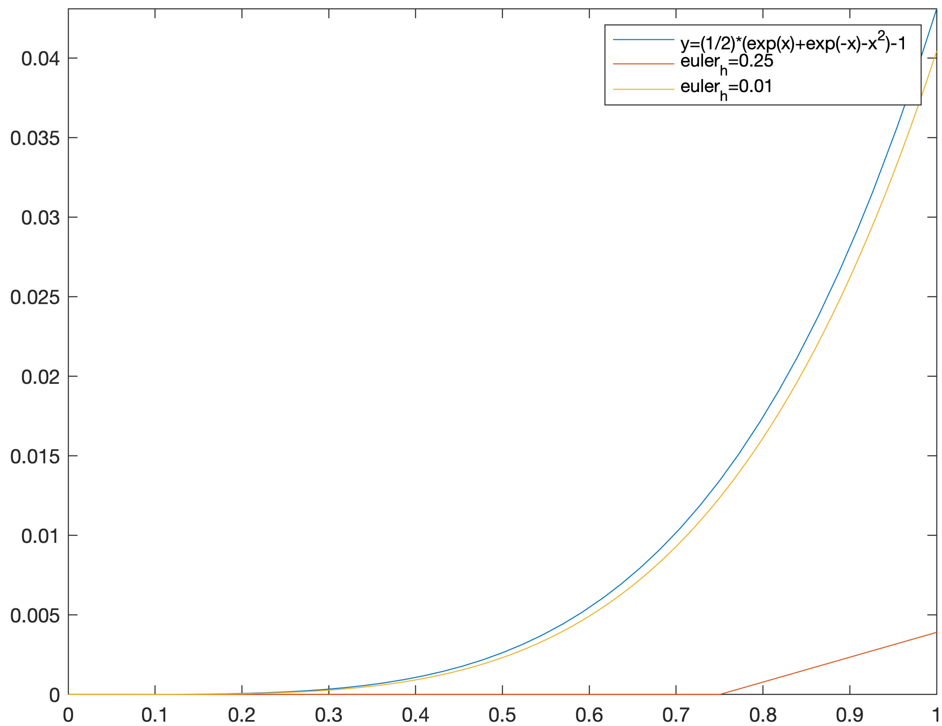
$y_3 = y'' \qquad y_3' = y_2 + t$

Numerical Analysis

Problem 2

c

```
clear
syms x
% The real function
f(x) = (1/2)*(exp(x) + exp(-x) - x^2) -1;

% The problem ask to use h =1/4,
% However, I see it is not well approximated
% So, I also plot a curve with h = 1/100
% To show how powerful the Euler method is

% Approximation with h = 1/4
[t4, y4] = euler_2c([0 1], [0 0 0], 4);
% Approximation with h = 1/100
[t100, y100] = euler_2c([0 1], [0 0 0], 100);

% Plotting on [0 1]
% Plot the function
fplot(f, [0 1])
hold on
% Plot for h = 1/4
plot(t4, y4(:,1))
% Plot for h = 1/100
plot(t100, y100(:,1))
hold off
legend("y=(1/2)*(exp(x)+exp(-x)-x^2)-1", "euler_h=0.25","euler_h=0.01")
```

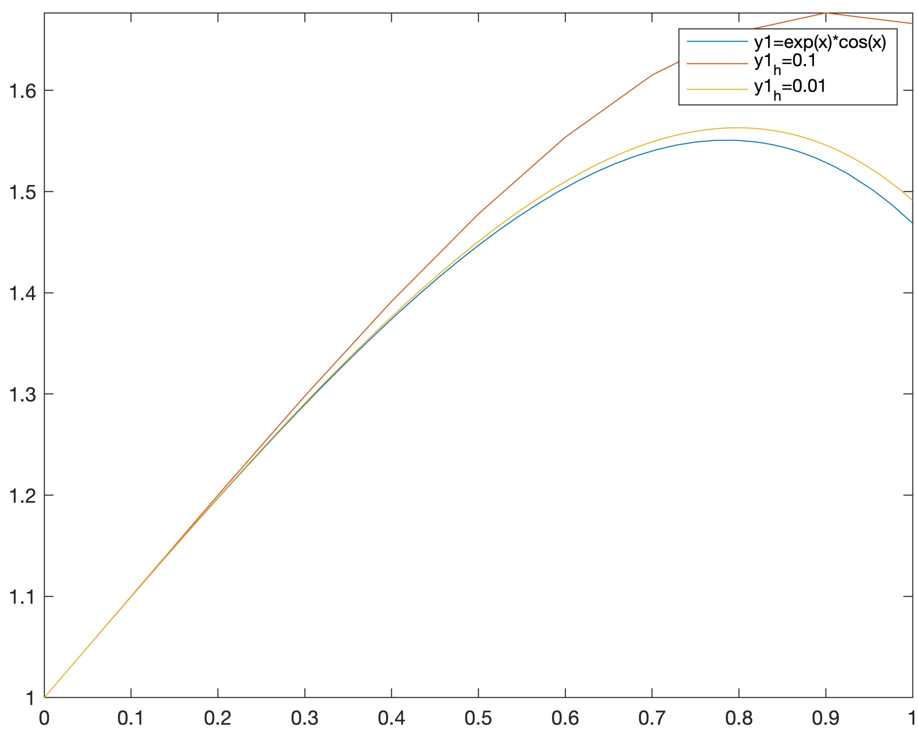


Problem 3

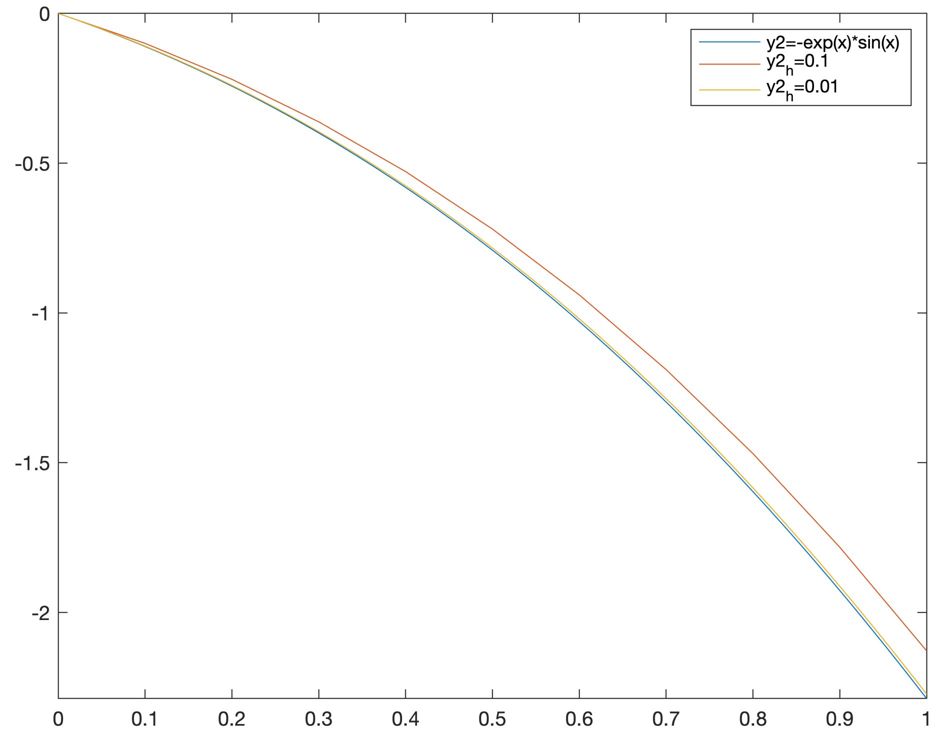
a

```
% Matlab functions at the end
syms x
y1(x) =  exp(x)*cos(x);
y2(x) = -exp(x)*sin(x);

% h=0.1
[th10, yh10] = euler_3a([0 1], [1 0], 10);

% h=0.01
[th100, yh100] = euler_3a([0 1], [1 0], 100);

%Plot y1
fplot(y1, [0 1]);
hold on
plot(th10, yh10(:,1));
plot(th100, yh100(:,1));
legend("y1=exp(x)*cos(x)", "y1_h=0.1", "y1_h=0.01")
```

```
hold off
```




```
%Plot y2
fplot(y2, [0 1]);
hold on
plot(th10, yh10(:,2));
plot(th100, yh100(:,2));
legend("y2=-exp(x)*sin(x)", "y2_h=0.1", "y2_h=0.01")
hold off
```




```
% global truncation error
% for h = 0.1
e10 = round(sqrt((yh10(end,1) - y1(1))^2 + (yh10(end,2) - y2(1))^2),5);
e10
```

e10 = 0.25355

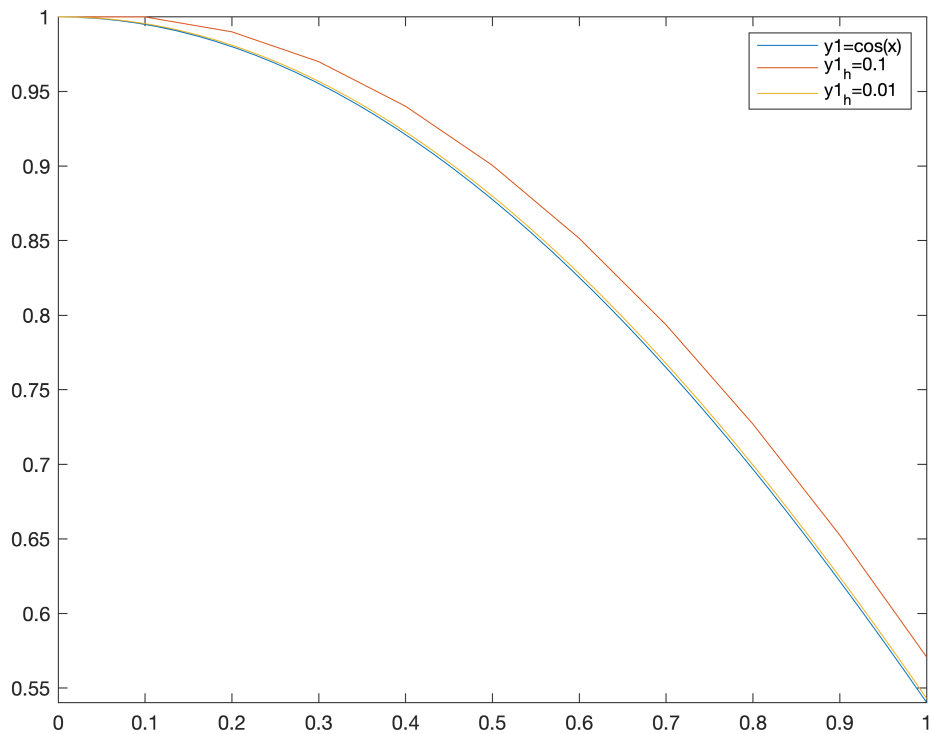
```
% for h = 0.01
e100 = round(sqrt((yh100(end,1) - y1(1))^2 + (yh100(end,2) - y2(1))^2),5);
e100
```

e100 = 0.027

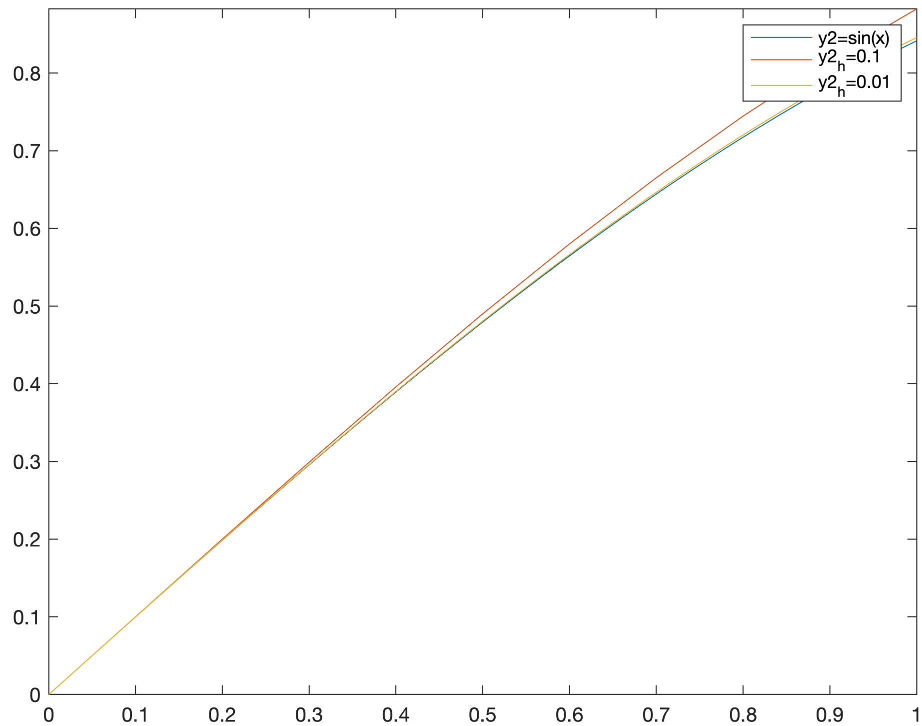
b

```
% Matlab functions at the end
syms x
y1(x) = cos(x);
y2(x) = sin(x);

% h=0.1
[th10, yh10] = euler_3b([0 1], [1 0], 10);
```

```
% h=0.01
[th100, yh100] = euler_3b([0 1], [1 0], 100);

%Plot y1
fplot(y1, [0 1]);
hold on
plot(th10, yh10(:,1));
plot(th100, yh100(:,1));
legend("y1=cos(x)", "y1_h=0.1", "y1_h=0.01")
hold off
```




```
%Plot y2
fplot(y2, [0 1]);
hold on
plot(th10, yh10(:,2));
plot(th100, yh100(:,2));
legend("y2=sin(x)", "y2_h=0.1", "y2_h=0.01")
hold off
```




```
% global truncation error
% for h = 0.1
e10 = round(sqrt((yh10(end,1) - y1(1))^2 + (yh10(end,2) - y2(1))^2),5);
e10
```

e10 = 0.05112

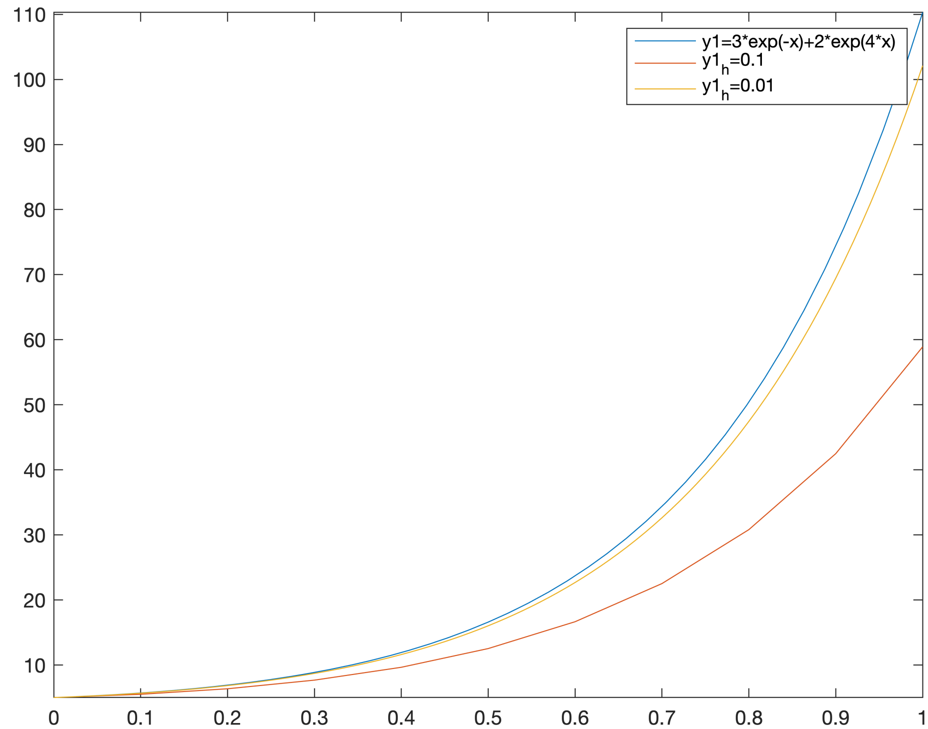
```
% for h = 0.01
e100 = round(sqrt((yh100(end,1) - y1(1))^2 + (yh100(end,2) - y2(1))^2),5);
e100
```

e100 = 0.00501

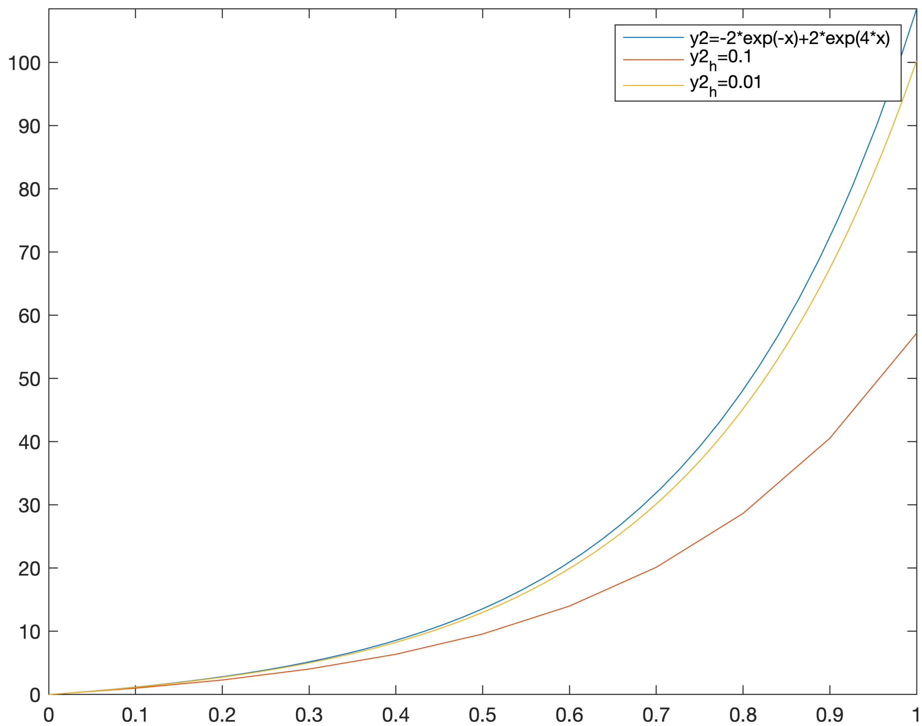
C

```
% Matlab functions at the end
syms x
y1(x) = 3*exp(-x)+2*exp(4*x);
y2(x) = -2*exp(-x)+2*exp(4*x);

% h=0.1
[th10, yh10] = euler_3c([0 1], [5 0], 10);
```

```
% h=0.01
[th100, yh100] = euler_3c([0 1], [5 0], 100);

%Plot y1
fplot(y1, [0 1]);
hold on
plot(th10, yh10(:,1));
plot(th100, yh100(:,1));
legend("y1=3*exp(-x)+2*exp(4*x)", "y1_h=0.1", "y1_h=0.01")
hold off
```




```
%Plot y2
fplot(y2, [0 1]);
hold on
plot(th10, yh10(:,2));
plot(th100, yh100(:,2));
legend("y2=-2*exp(-x)+2*exp(4*x)", "y2_h=0.1", "y2_h=0.01")
hold off
```




```
% global truncation error
% for h = 0.1
e10 = round(sqrt((yh10(end,1) - y1(1))^2 + (yh10(end,2) - y2(1))^2),5);
e10
```

e10 = 72.62693

```
% for h = 0.01
e100 = round(sqrt((yh100(end,1) - y1(1))^2 + (yh100(end,2) - y2(1))^2),5);
e100
```

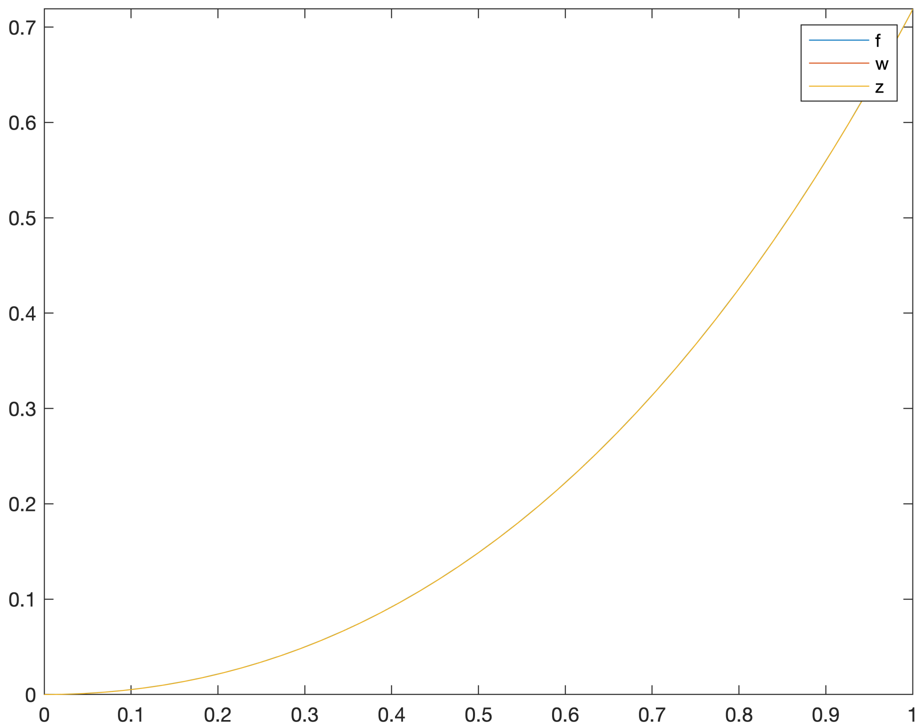
e100 = 11.57863

Problem 4

a) y' = t + y

```
% initial condition y(0) = 0
syms x
f(x) = exp(x) - x - 1; % solve the inital value problem
[t, w, z, step, hmax] = rk23_4a([0 1] ,0 ,10^(-8));
fplot(f, [0 1])
```

```
hold on
plot(t, w)
plot(t, z)
legend("f", "w", "z")
hold off
```




```
%Number of steps
step
```

```
step = 879
```

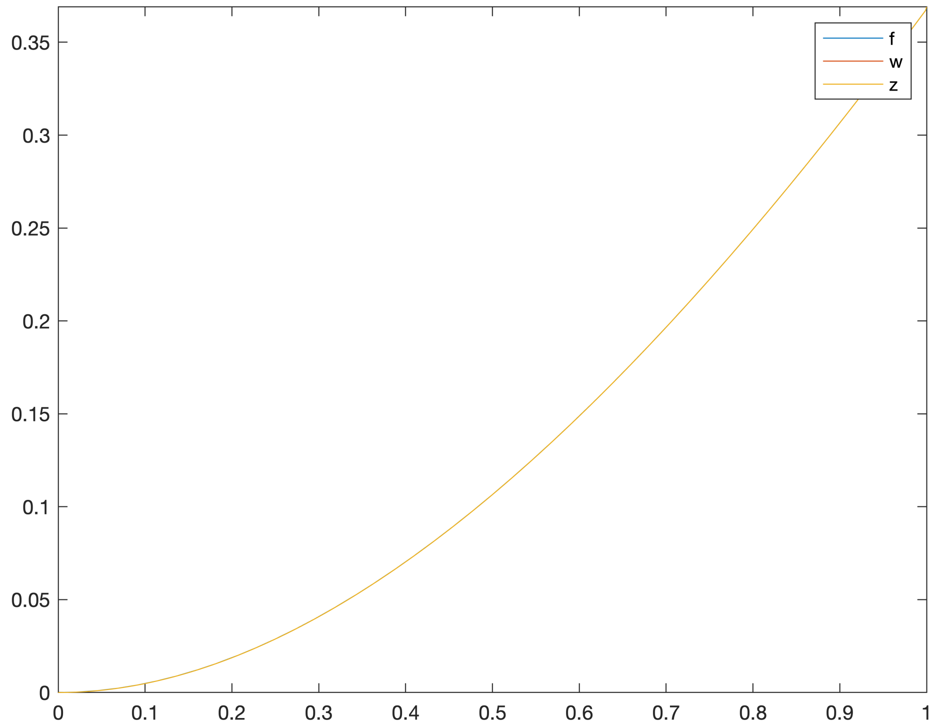
```
% max step size
hmax
```

```
hmax = 0.0100
```

b) y' = t - y

```

% initial condition y(0) = 0
syms x
f(x) = exp(-x) + x - 1; % solve the inital value problem
[t, w, z, step, hmax] = rk23_4b([0 1] ,0 ,10^(-8));
fplot(f, [0 1])
hold on
plot(t, w)
plot(t, z)
```

```
legend("f", "w", "z")
hold off
```




```
%Number of steps
step
```

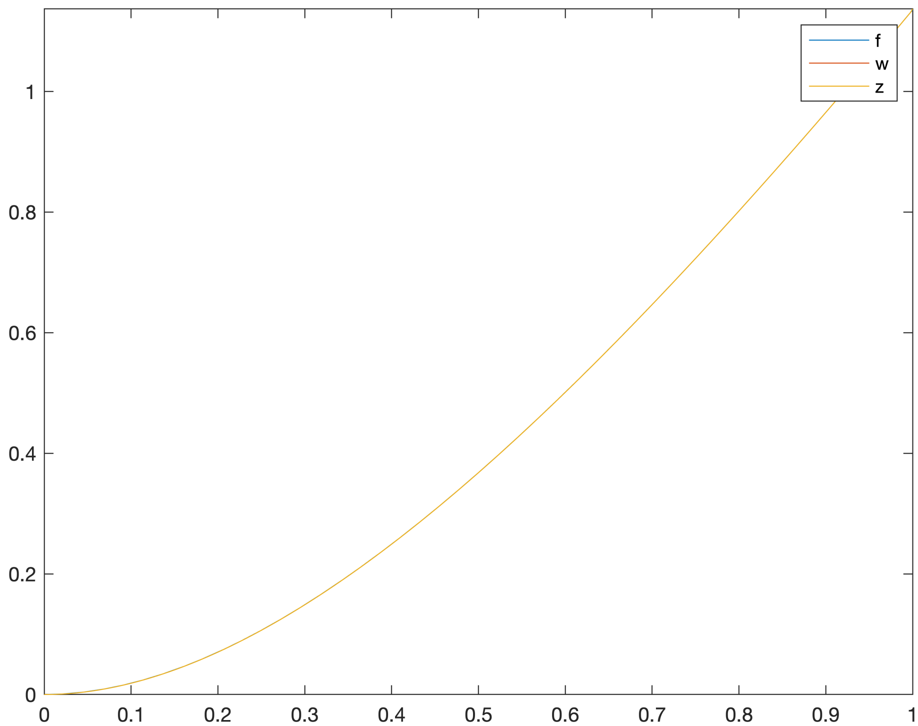
```
step = 748
```

```
% max step size
hmax
```

```
hmax = 0.0100
```

c) y' = 4t - 2y

```
% initial condition y(0) = 0
syms x
f(x) = exp(-2*x) + 2*x - 1; % solve the inital value problem
[t, w, z, step, hmax] = rk23_4c([0 1] ,0 ,10^(-8));
fplot(f, [0 1])
hold on
plot(t, w)
plot(t, z)
legend("f", "w", "z")
hold off
```




```
%Number of steps
step
```

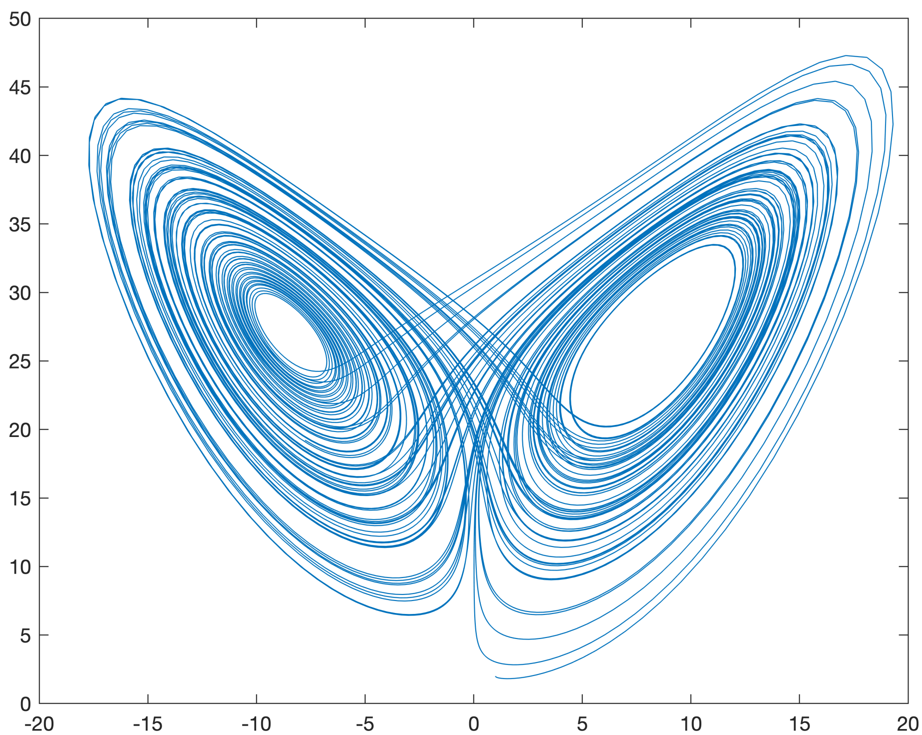
```
step = 866
```

```
% max step size
hmax
```

```
hmax = 0.0100
```

Problem 5

```
[t, y] = rk4_p5([0 100] ,[1 1 2] ,10000);
plot(y(:,1), y(:,3)) %Plot x-z as in the slides
```



functions

```
function [t, y] = euler_2c(inter ,ic ,n)
% Euler methods for 2c
h=(inter(2)-inter (1))/n; % plot n points in total
y(1,:)=ic; % enter initial conds in y
t(1)=inter(1);

for k=1:n
t(k+1)=t(k)+h;
y(k+1,:)=eulerstep_2c(t(k),y(k,:),h);
end
end

function y=eulerstep_2c(t,y,h)
%one step of Euler's Method
%Input: current time t, current value y, stepsize h
%Output: approximate solution value at time t+h
y=y+h*ydot_2c(t,y);
end

function z=ydot_2c(t,y)
%right-hand function z=ydot(t,y)
%right-hand side of differential equation
```

```
z(1)=y(2);
z(2)=y(3);
z(3)=y(2) + t;
end

% Euler approximation for 3a
function [t, y] = euler_3a(inter ,ic ,n)
h=(inter(2)-inter (1))/n; % plot n points in total
y(1,:)=ic; % enter initial conds in y
t(1)=inter(1);

for k=1:n
t(k+1)=t(k)+h;
y(k+1,:)=eulerstep_3a(t(k),y(k,:),h);
end
end

function y=eulerstep_3a(t,y,h)
%one step of Euler’s Method
%Input: current time t, current value y, stepsize h
%Output: approximate solution value at time t+h
y=y+h*ydot_3a(t,y);
end

function z=ydot_3a(t,y)
%right-hand function z=ydot(t,y)
%right-hand side of differential equation
z(1)=y(1) + y(2);
z(2)=-y(1) + y(2);
end

% Euler approximation for 3b
function [t, y] = euler_3b(inter ,ic ,n)
h=(inter(2)-inter (1))/n; % plot n points in total
y(1,:)=ic; % enter initial conds in y
t(1)=inter(1);

for k=1:n
t(k+1)=t(k)+h;
y(k+1,:)=eulerstep_3b(t(k),y(k,:),h);
end
end

function y=eulerstep_3b(t,y,h)
%one step of Euler’s Method
%Input: current time t, current value y, stepsize h
%Output: approximate solution value at time t+h
y=y+h*ydot_3b(t,y);
end
```

```
function z=ydot_3b(t,y)
%right-hand function z=ydot(t,y)
%right-hand side of differential equation
z(1)= -y(2);
z(2)=  y(1);
end

% Euler approximation for 3c
function [t, y] = euler_3c(inter ,ic ,n)
h=(inter(2)-inter (1))/n; % plot n points in total
y(1,:)=ic; % enter initial conds in y
t(1)=inter(1);

for k=1:n
t(k+1)=t(k)+h;
y(k+1,:)=eulerstep_3c(t(k),y(k,:),h);
end
end

function y=eulerstep_3c(t,y,h)
%one step of Euler's Method
%Input: current time t, current value y, stepsize h
%Output: approximate solution value at time t+h
y=y+h*ydot_3c(t,y);
end

function z=ydot_3c(t,y)
%right-hand function z=ydot(t,y)
%right-hand side of differential equation
z(1)= y(1) + 3*y(2);
z(2)= 2*y(1) + 2*y(2);
end

%
% RK23 for 4a
function [t, w, z, step, hmax] = rk23_4a(inter ,ic ,tol)
w(1,:)=ic; % enter initial conds in y
z(1,:)=ic;
t(1)=inter(1);
trial = 0; % The indicator to set h* = h_k or h* = h_(k+1)
h = 0.01;
hmax = h;
k = 1;

while t(end) ~= 1
[w(k+1,:), z(k+1,:), h]=rk23step_4a(t(k),w(k,:),h);
t(k+1)=min(t(k)+h, 1);

e_relative = abs(w(k+1)-z(k+1))/abs(w(k+1));
% step size h
```

```
% implemented follow T.Sauer page 341
if  e_relative < tol % goal achieved
    trial = 0;
    h = 0.8*h*(tol/e_relative)^(1/3);
    if h > hmax
        hmax = h;
    end

elseif e_relative > tol && trial == 0
    % goal not achieved
    % give it another try
    h = h;
    trial = 1;
elseif e_relative > tol && trial == 1
    % goal is not achieved 2nd try
    % cut step size in half
    h = h/2;
    trial = 0;
end
k = k+1;
end
step = length(t);
end

function [w,z, h]=rk23step_4a(t,w,h)
%one step of the Runge-Kutta order 2 3 method
s1=ydot_4a(t,w);
s2=ydot_4a(t+h,w+h*s1);
s3=ydot_4a(t+h/2,w+h*(s1 + s2)/4);
z=w+h*(s1+4*s3+s2)/6;
w=w+h*(s1+s2)/2;
end

function z=ydot_4a(t,y)
z=t + y;
end

%
% RK23 for 4b
function [t, w, z, step, hmax] = rk23_4b(inter ,ic ,tol)
w(1,:)=ic; % enter initial conds in y
z(1,:)=ic;
t(1)=inter(1);
trial = 0; % The indicator to set h* = h_k or h* = h_(k+1)
h = 0.01;
hmax = h;
k = 1;

while t(end) ~= 1
[w(k+1,:), z(k+1,:), h]=rk23step_4b(t(k),w(k,:),h);
```

```
t(k+1)=min(t(k)+h, 1);

e_relative = abs(w(k+1)-z(k+1))/abs(w(k+1));
% step size h
% implemented follow T.Sauer page 341
if  e_relative < tol % goal achieved
    trial = 0;
    h = 0.8*h*(tol/e_relative)^(1/3);
    if h > hmax
        hmax = h;
    end

elseif e_relative > tol && trial == 0
    % goal not achieved
    % give it another try
    h = h;
    trial = 1;
elseif e_relative > tol && trial == 1
    % goal is not achieved 2nd try
    % cut step size in half
    h = h/2;
    trial = 0;
end
k = k+1;
end
step = length(t);
end

function [w,z, h]=rk23step_4b(t,w,h)
%one step of the Runge-Kutta order 2 3 method
s1=ydot_4b(t,w);
s2=ydot_4b(t+h,w+h*s1);
s3=ydot_4b(t+h/2,w+h*(s1 + s2)/4);
z=w+h*(s1+4*s3+s2)/6;
w=w+h*(s1+s2)/2;
end

function z=ydot_4b(t,y)
z=t - y;
end

%
% RK23 for 4c
function [t, w, z, step, hmax] = rk23_4c(inter ,ic ,tol)
w(1,:)=ic; % enter initial conds in y
z(1,:)=ic;
t(1)=inter(1);
trial = 0; % The indicator to set h* = h_k or h* = h_(k+1)
h = 0.01;
hmax = h;
```

```
k = 1;

while t(end) ~= 1
[w(k+1,:), z(k+1,:), h]=rk23step_4c(t(k),w(k,:),h);
t(k+1)=min(t(k)+h, 1);

e_relative = abs(w(k+1)-z(k+1))/abs(w(k+1));
% step size h
% implemented follow T.Sauer page 341
if  e_relative < tol % goal achieved
    trial = 0;
    h = 0.8*h*(tol/e_relative)^(1/3);
    if h > hmax
        hmax = h;
    end

elseif e_relative > tol && trial == 0
    % goal not achieved
    % give it another try
    h = h;
    trial = 1;
elseif e_relative > tol && trial == 1
    % goal is not achieved 2nd try
    % cut step size in half
    h = h/2;
    trial = 0;
end
k = k+1;
end
step = length(t);
end

function [w,z, h]=rk23step_4c(t,w,h)
%one step of the Runge-Kutta order 2 3 method
s1=ydot_4c(t,w);
s2=ydot_4c(t+h,w+h*s1);
s3=ydot_4c(t+h/2,w+h*(s1 + s2)/4);
z=w+h*(s1+4*s3+s2)/6;
w=w+h*(s1+s2)/2;
end

function z=ydot_4c(t,y)
z=4*t - 2*y;
end

%RK4 for 5
function [t, y] = rk4_p5(inter ,ic ,n)
h=(inter(2)-inter (1))/n; % plot n points in total
y(1,:)=ic; % enter initial conds in y
t(1)=inter(1);
```

```
for k=1:n
t(k+1)=t(k)+h;
y(k+1,:)=rk4step_p5(t(k),y(k,:),h);
end
end

function y=rk4step_p5(t,w,h)
%one step of the Runge-Kutta order 4 method
s1=ydot_p5(t,w);
s2=ydot_p5(t+h/2,w+h*s1/2);
s3=ydot_p5(t+h/2,w+h*s2/2);
s4=ydot_p5(t+h,w+h*s3);
y=w+h*(s1+2*s2+2*s3+s4)/6;
end

function z=ydot_p5(t,y)
%Lorenz equations
p=10; q=8/3;r=28;
z(1)=p*(y(2) - y(1));
z(2)=r*y(1) - y(2) - y(1)*y(3);
z(3)=y(1)*y(2)-q*y(3);
end
```