Lab 2

Souradeep Bhattacharya 861105938
EE105 Section: 024

January 25, 2018

Prelab

Given the transfer function:

$$H(s) = \frac{25}{s^2 + 4s + 25}$$

We can determine the following values:

$$\begin{aligned}
\omega_n &= 5 \\
G &= 1 \\
\zeta &= \frac{2}{5} \\
\sigma &= \zeta\omega_n = 2 \\
\omega_d &= \omega_n\sqrt{1-\zeta} = 3.8729
\end{aligned}$$

The steady state response is given by:

$$\begin{aligned}
H(s) &= \frac{25}{s^2 + 4s + 25} \\
H(s)|_{s=j\omega} &= \frac{25}{25 - \omega^2 + 4j\omega} \\
|H(j\omega)|^2 &= H(j\omega)H(j\omega)^* \\
&= \left(\frac{25}{25 - \omega^2 + 4j\omega}\right)\left(\frac{25}{25 - \omega^2 - 4j\omega}\right) \\
|H(j\omega)| &= \frac{25}{\sqrt{((25-\omega^2)^2 + 16\omega^2)}} \\
\angle H(j\omega) &= \left(\frac{25}{25 - \omega^2 + 4j\omega}\right)\left(\frac{25 - \omega^2 - 4j\omega}{25 - \omega^2 - 4j\omega}\right) \\
&= \frac{25(25 - \omega^2 - 4j\omega)}{(25-\omega^2)^2 - 16\omega^2}
\end{aligned}$$

The steady state response is given by the following:

$$y(t) = 0.999sin(0.1t - 0.9163°)$$

Given that $x = \begin{bmatrix} y & \dot{y} \end{bmatrix}^T$

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= \ddot{y}(t) \\
\frac{Y(s)}{U(s)} &= \frac{25}{s^2 + 4s + 25} \\
Y(s)\left(s^2 + 4s + 25\right) &= 25U(s) \\
s^2Y(s) + 4sY(s) + 25Y(s) &= 25U(s) \\
\mathcal{L}^{-1}\left[s^2Y(s) + 4sY(s) + 25Y(s)\right] &= \mathcal{L}^{-1}\left[25U(s)\right] \\
\ddot{y}(t) + 4\dot{y}(t) + 25y(t) &= 25u(t) \\
\ddot{y}(t) &= 25u(t) - 4\dot{y}(t) - 25y(t) \\
\ddot{y}(t) &= 25u(t) - 4x_2 - 25x_1 \\
\dot{x}_2 &= 25u(t) - 4x_2 - 25x_1 \\
\dot{x} &= \begin{bmatrix} x_2 \\ 25u(t) - 4x_2 - 25x_1 \end{bmatrix} \\
y &= \begin{bmatrix} x_1 \end{bmatrix}
\end{aligned}$$

The system is linear, therefore we can solve for A, B, C, D

$$A = \begin{bmatrix} 0 & 1 \\ -25 & -4 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ 25 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad D = \begin{bmatrix} 0 \end{bmatrix}$$

Lab Results

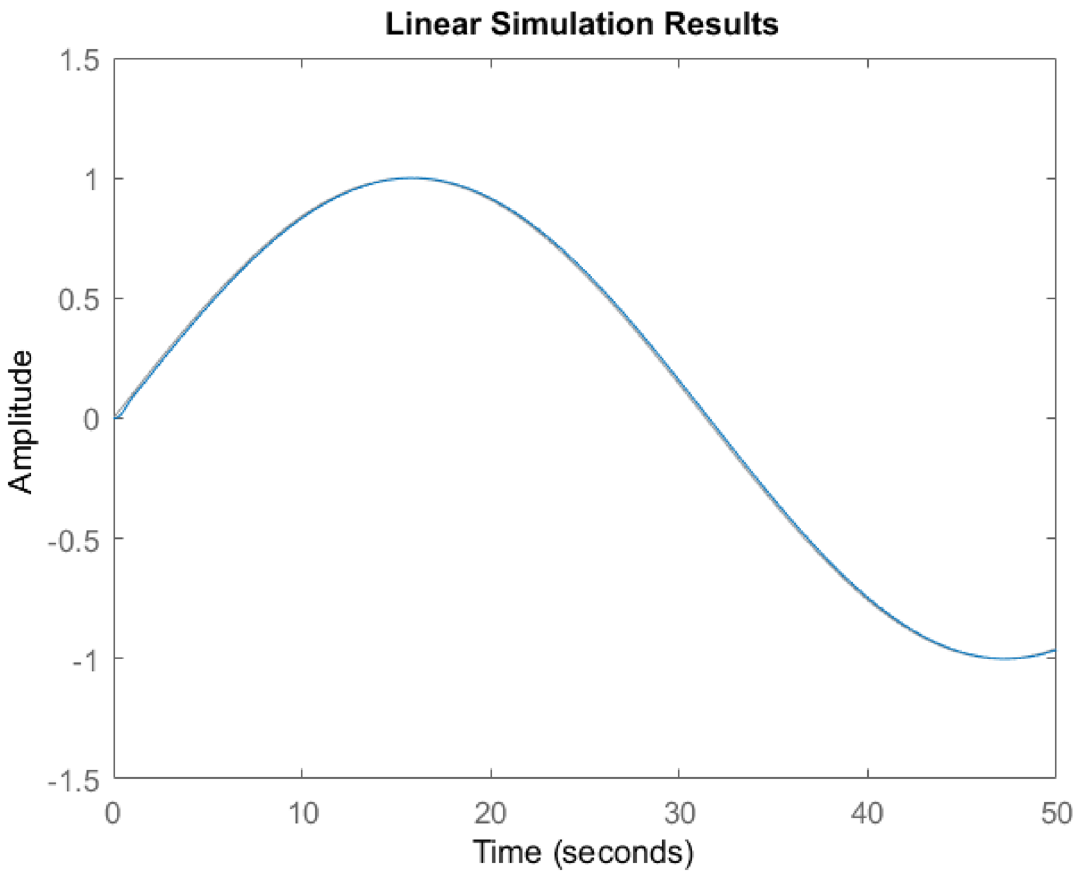

Figure 1: Results of lsim using $u(t) = sin(0.1t)$

In the above figure we can see the results of using the lsim command in Matlab. Upon closer inspection we can see that there is a slight phase shift in the output which is what we expected.

```
function [ dx ] = f( t,x,u )
%f This function does the one step of the simulation

dx = [0 1;-25 -4]*x + [0;25]*u;

end
```

The above function was responsible for doing one step of the simulation. The following code was responsible for calling this function over and over again conducting the Euler's simulation. It also generated the plots and the error figures.

```
x0=[3;0];
T=0.01;
N = round(50/T)+1;
t = zeros(1,N);
x = zeros(2,N);
x(:,1) = x0;
```

```
t = T*(0:N-1);
u = 0.*t;
%u = sin(0.1.*t);
%flops(0);

for i=1:N
   dx = f(t(i),x(:,i),u(i));
   x(:,i+1) = x(:,i) + dx*T;
end
%flops

clf;
y=x(1,1:i);
y2=x(2,1:i);

subplot(2,1,1);
plot(t,y);
str = sprintf('Euler Simulation (x1) for T = %g',T);
title(str);
xlabel('Time,t,seconds');
ylabel('Position,x1');

subplot(2,1,2);
plot(t,y2);
str = sprintf('Euler Simulation (x2) for T = %g',T);
title(str);
xlabel('Time,t,seconds');
ylabel('Velocity,x2');


tstr = sprintf('t_%g',T);
tstr = strrep(tstr, '.', '_');
filename = sprintf('es_%s',tstr);
print(filename,'-dpng');

%% Linsim
[y_l,time,x] = getLinSimResults(50,T,u,x0);

y1_l=x(:,1);
y2_l=x(:,2);

figure;
subplot(2,1,1)
plot(t,y1_l);
str = sprintf('Linsim (x1) result for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Position,y');

subplot(2,1,2)
plot(t,y2_l);
str = sprintf('Linsim (x2) result for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Velocity,y2');


filename = sprintf('ls_%s',tstr);
print(filename,'-dpng');
```

```


%% Error Calc
error = y-y1_l';
error2 = y2-y2_l';

figure;
subplot(2,1,1)
plot(t,error)
str = sprintf('Error between Eulers simulation and linsim (x1) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Error');

subplot(2,1,2)
plot(t,error2)
str = sprintf('Error between Eulers simulation and linsim (x2) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Error');


filename = sprintf('e_%s',tstr);
print(filename,'-dpng');
```

Selecting T

I first ran the program with T=0.5 to see what would happen. The result I obtained was very far from what I was expecting. (Figure below)

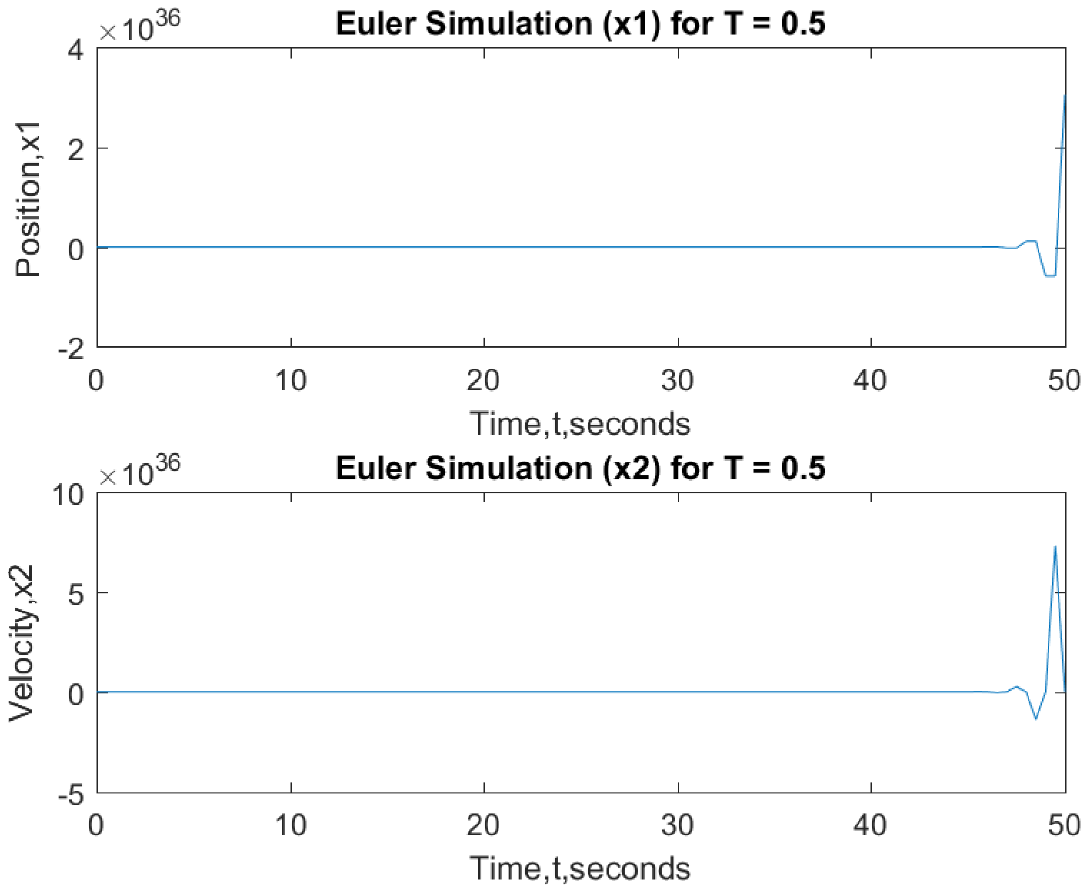

Figure 2: The results of the Euler simulation with just a initial condition and no input

This simulation took 0.0075 seconds to run. After setting T to 0.01 I got the following:

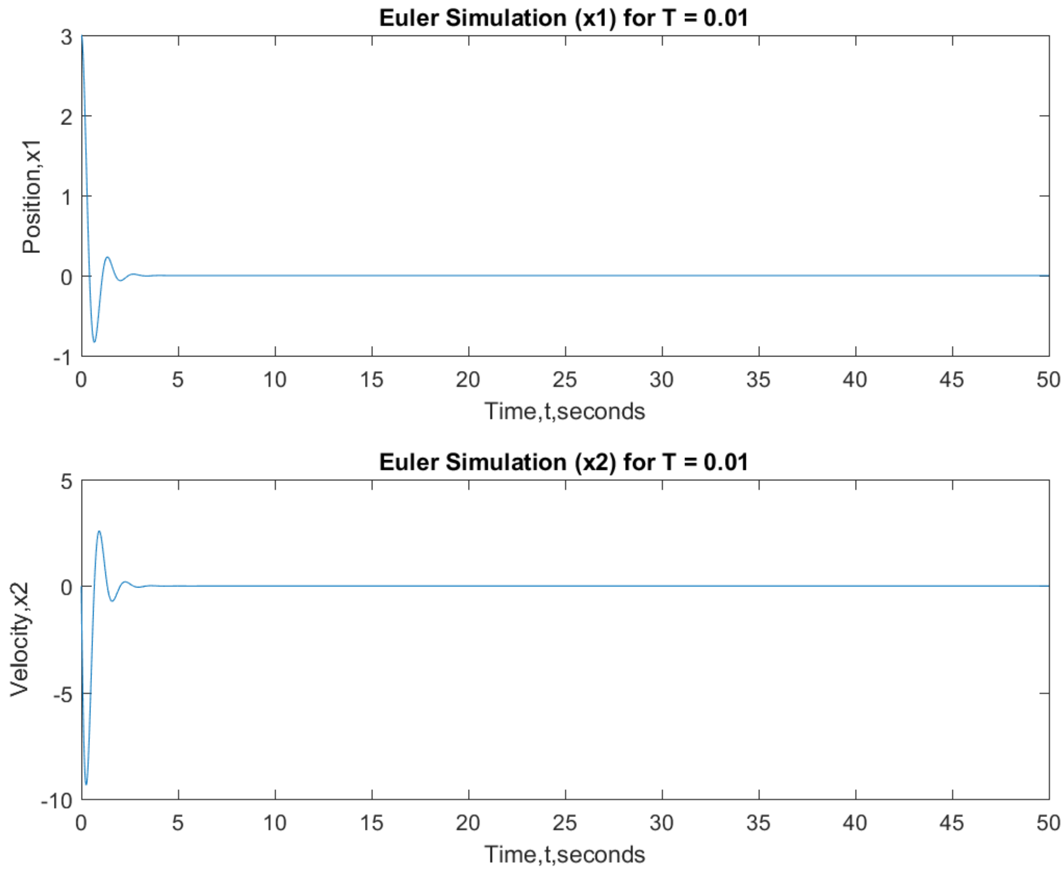

Figure 3: The results of the Euler simulation with just a initial condition and no input

This simulation took 0.02 seconds.

No Input and Initial Conditions = 0

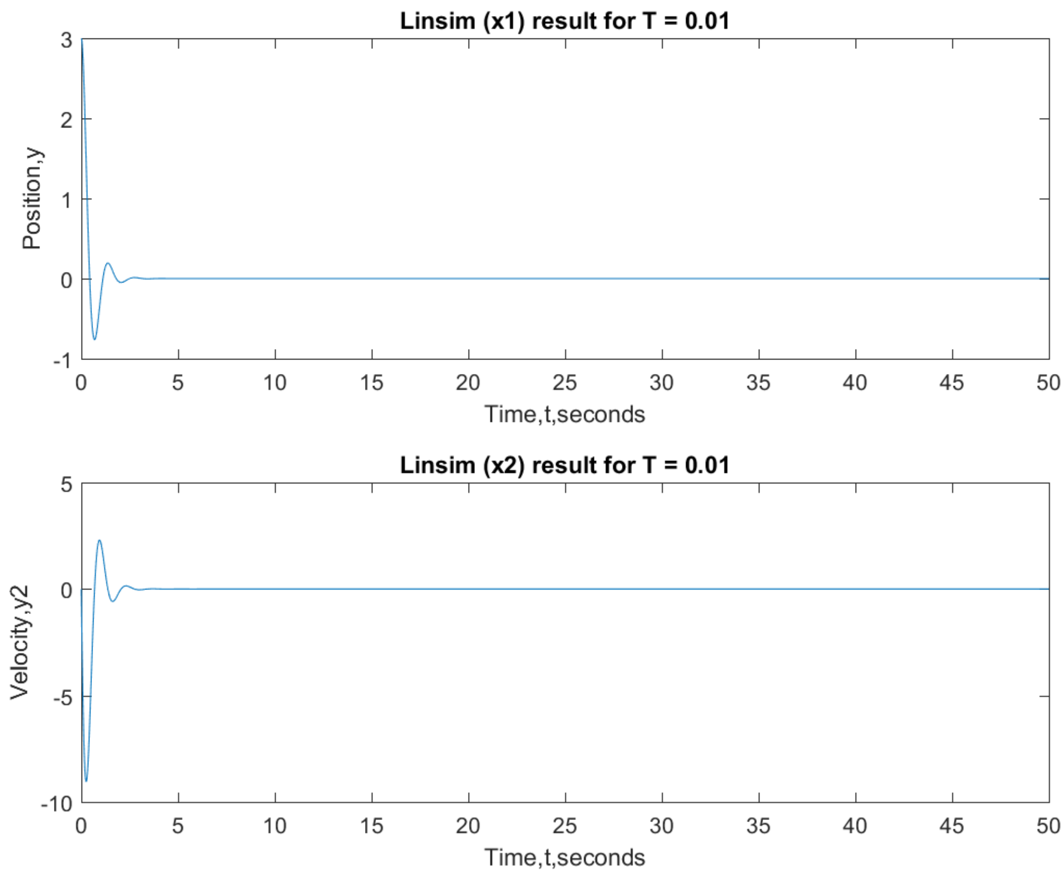

Figure 4: The results of the Linear simulation with just a initial condition and no input

The figures above show the results of the Euler simulation and the Linsim function. The error between them was calculated as $EulerSimuation - LinearSimulation$. This is shown in the figure below.

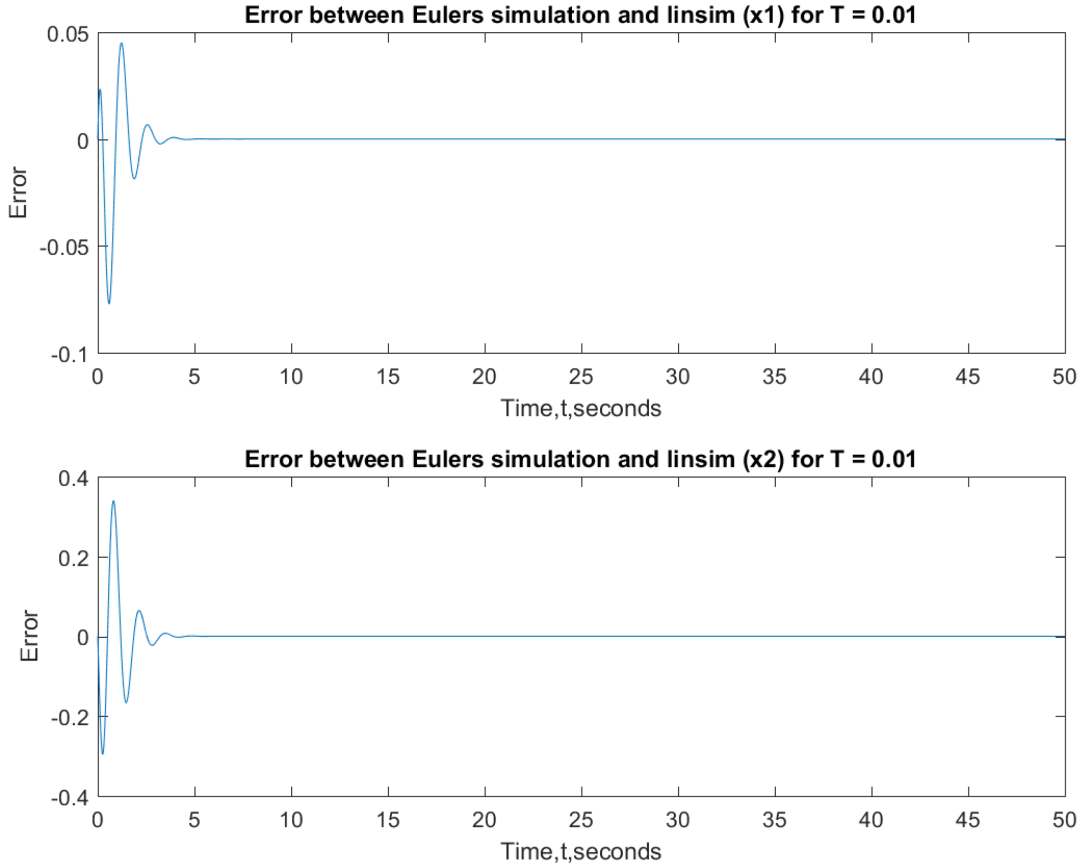

Figure 5: The results of the Euler simulation with just a initial condition and no input

As you can see the error between the two is extremely small and as a result the two match each other well.

With input and initial conditions = 0

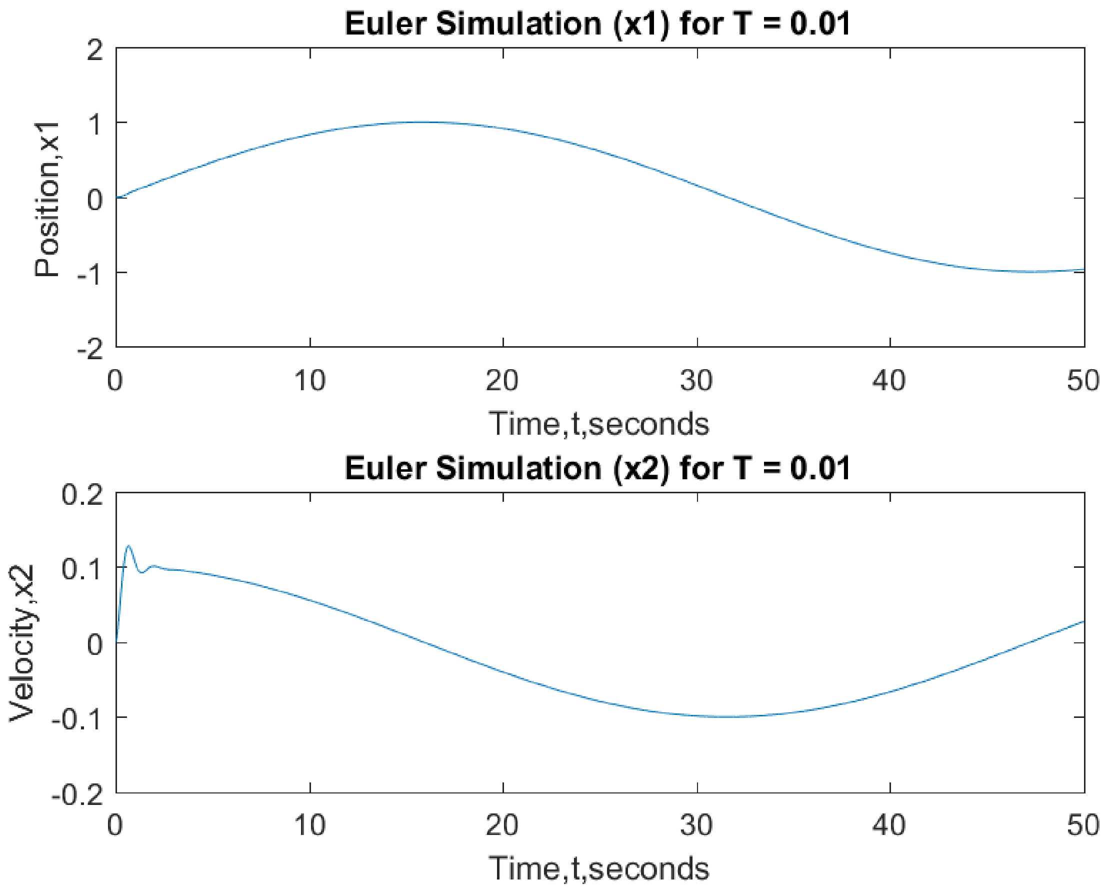

Figure 6: The results of the Euler simulation

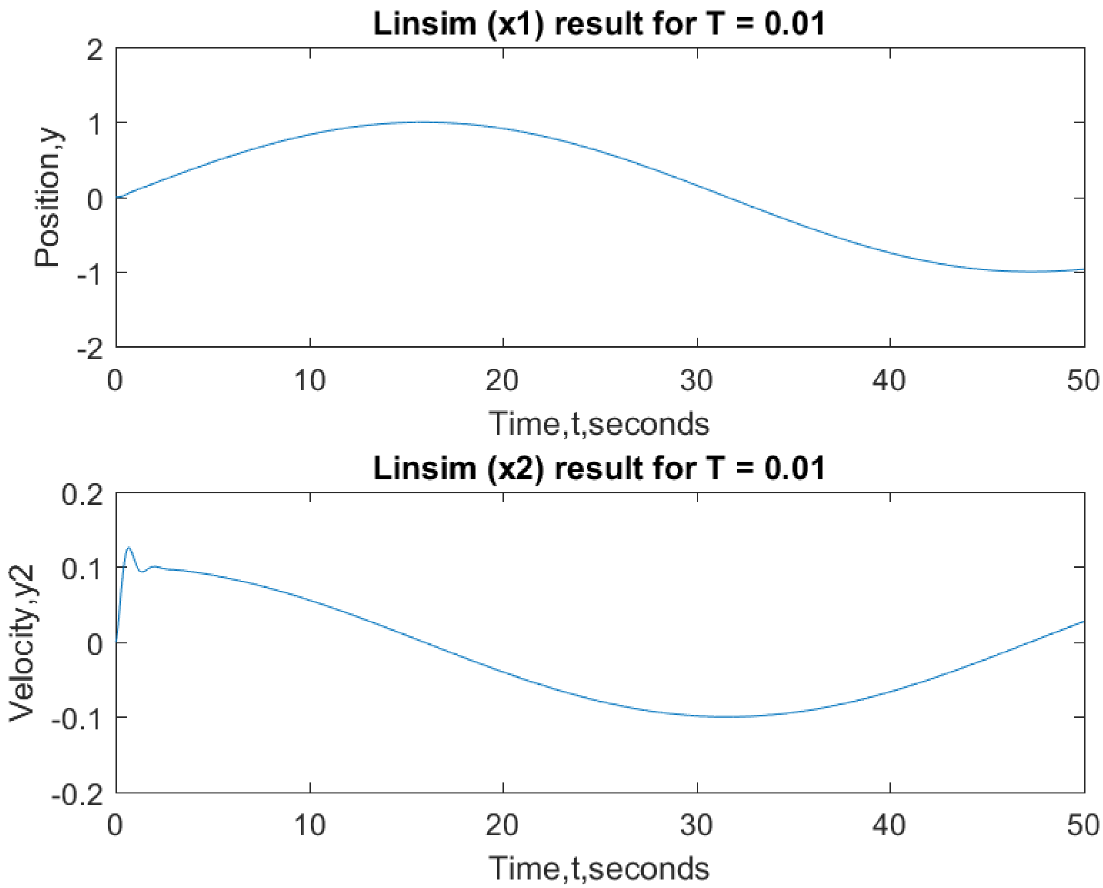

Figure 7: The results of the linear simulation

The figures above show the results of the Euler simulation and the Linsim function. The error between them was calculated as $EulerSimuation - LinearSimulation$. This is shown in the figure below.

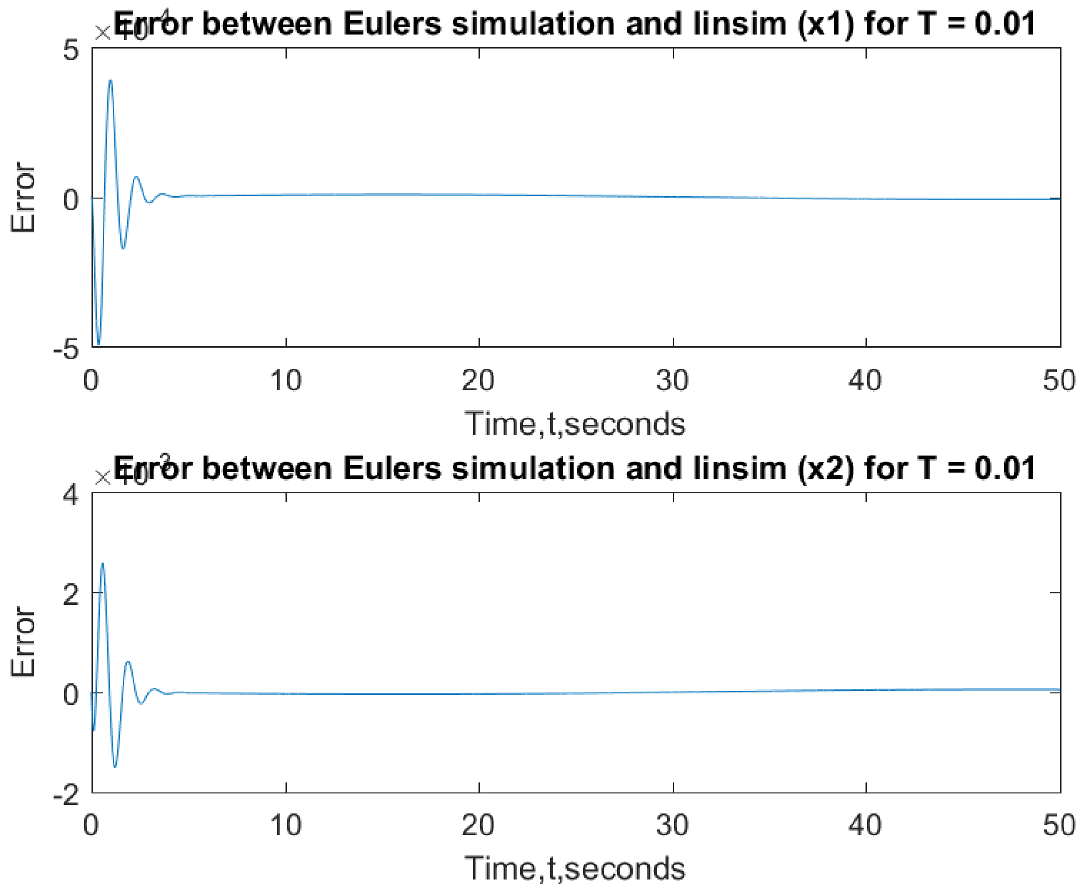

Figure 8: The difference between euler simulation and linsim

As you can see the error between the two is extremely small and as a result the two match each other well.

With input and initial conditions = [3;0]

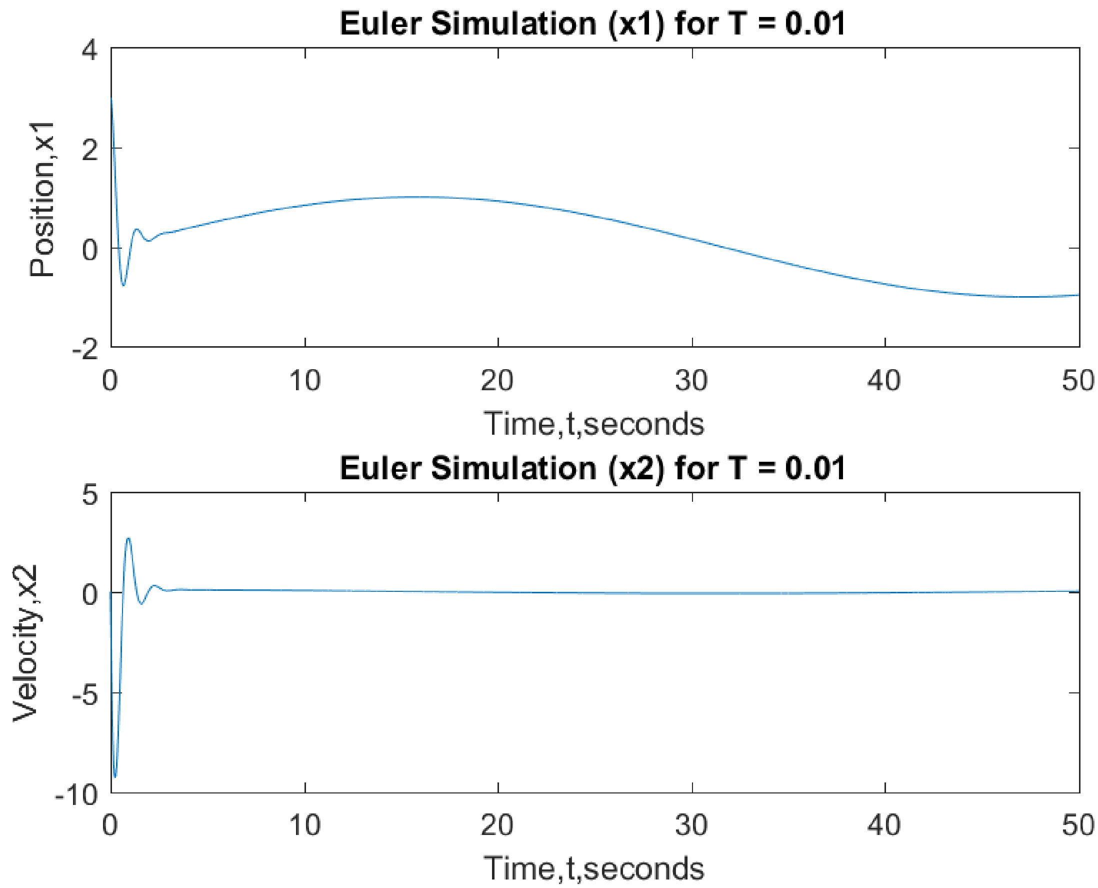

Figure 9: The results of the Euler simulation

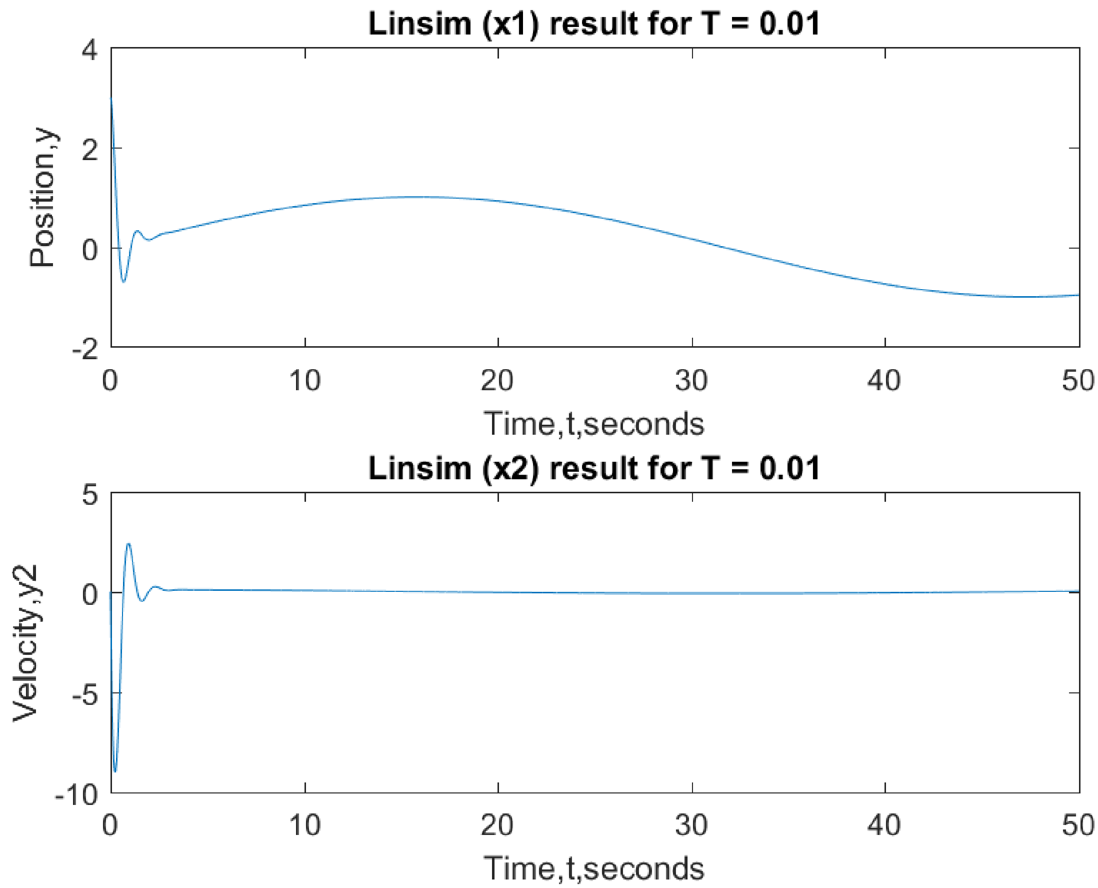

Figure 10: The results of the linear simulation

The figures above show the results of the Euler simulation and the Linsim function. The error between them was calculated as $EulerSimuation - LinearSimulation$. This is shown in the figure below.

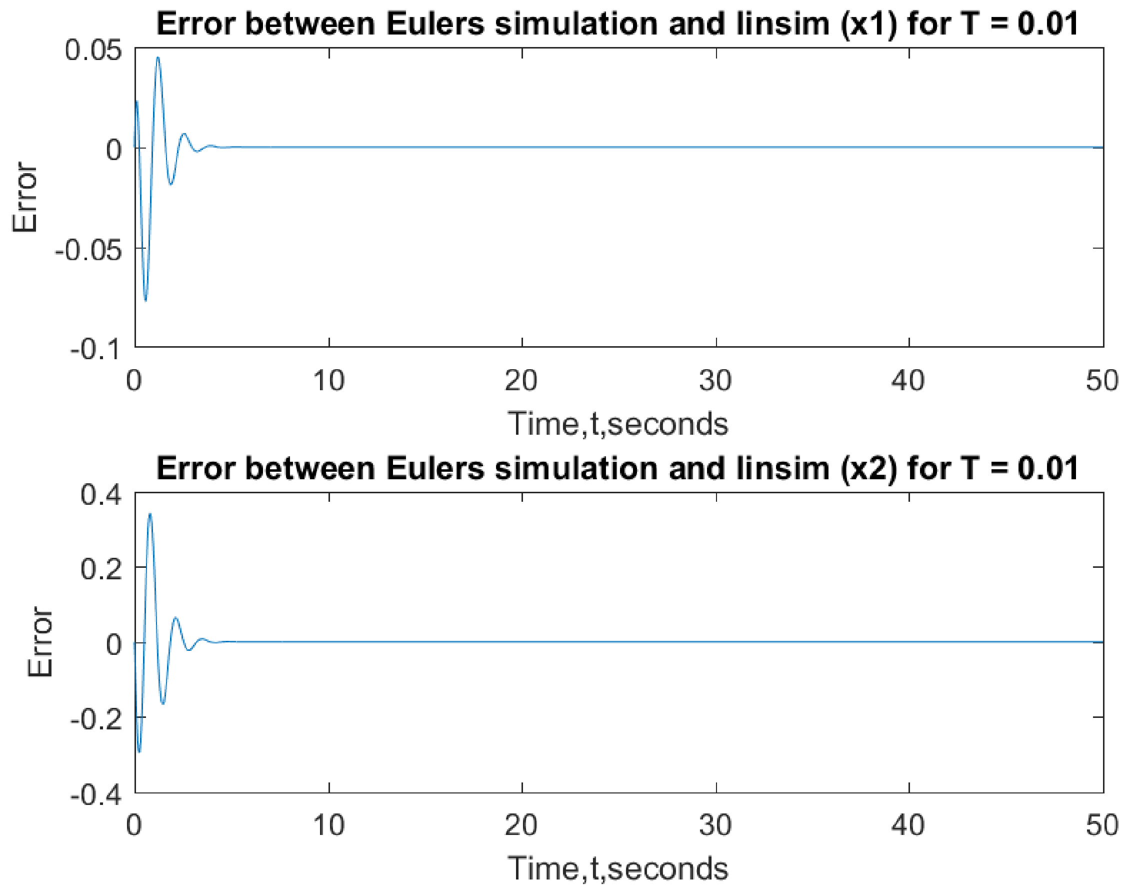

Figure 11: The error between the euler simulation and linsim

As you can see the error between the two is extremely small and as a result the two match each other well.

ODE23

Just running ODE23 in Matlab resulted in the following figure.

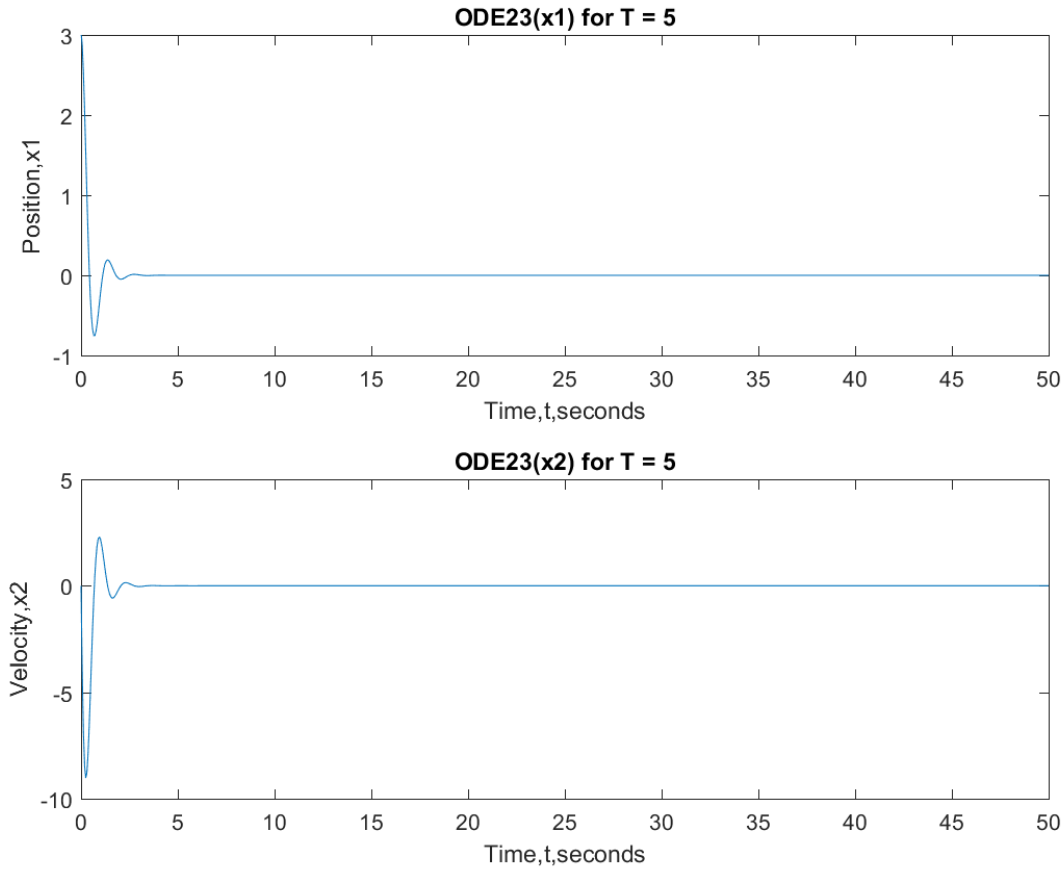

Figure 12: The results of the ODE23 with just a initial condition and no input

When comparing ODE23 with our Euler Simulation we obtain the following figure:

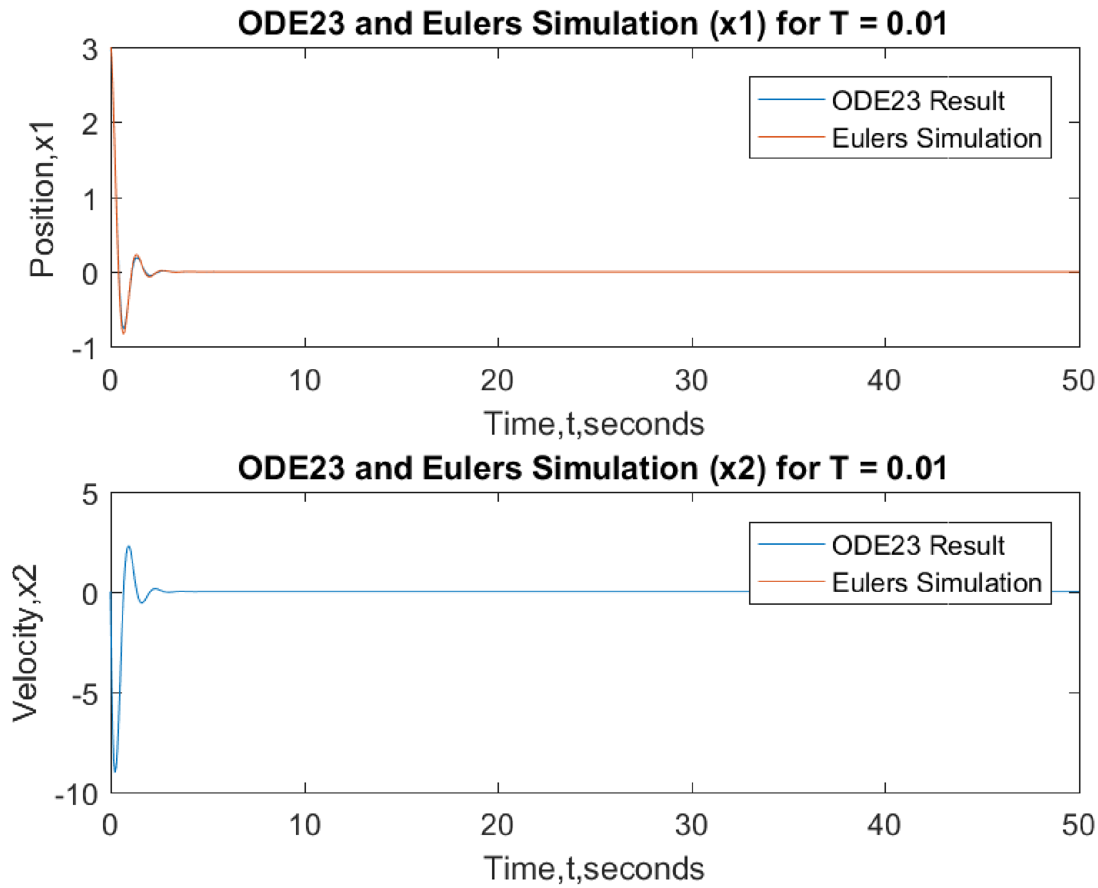

Figure 13: The results of the ODE23 overlaid with the Euler's Simulation.

We can see that the two line up extremely well. The ODE23 simulation took 0.09 seconds to run on my laptop. When we add the input back in we obtain the following:

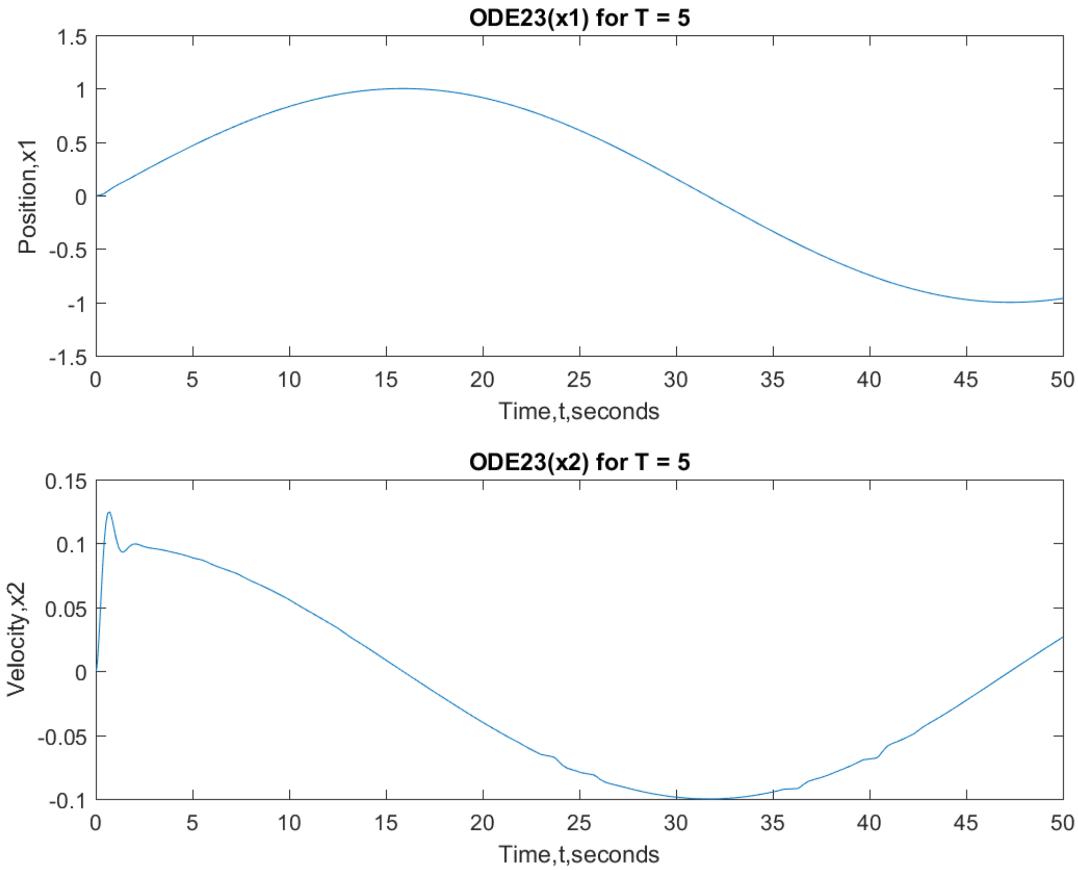

Figure 14: The results of the ODE23 simulation with an input of $u(t) = sin(0.1t)$

The error between the predicted result and the simulated result was calculated as $predicted - odesimulation$. This resulted in the following graph.

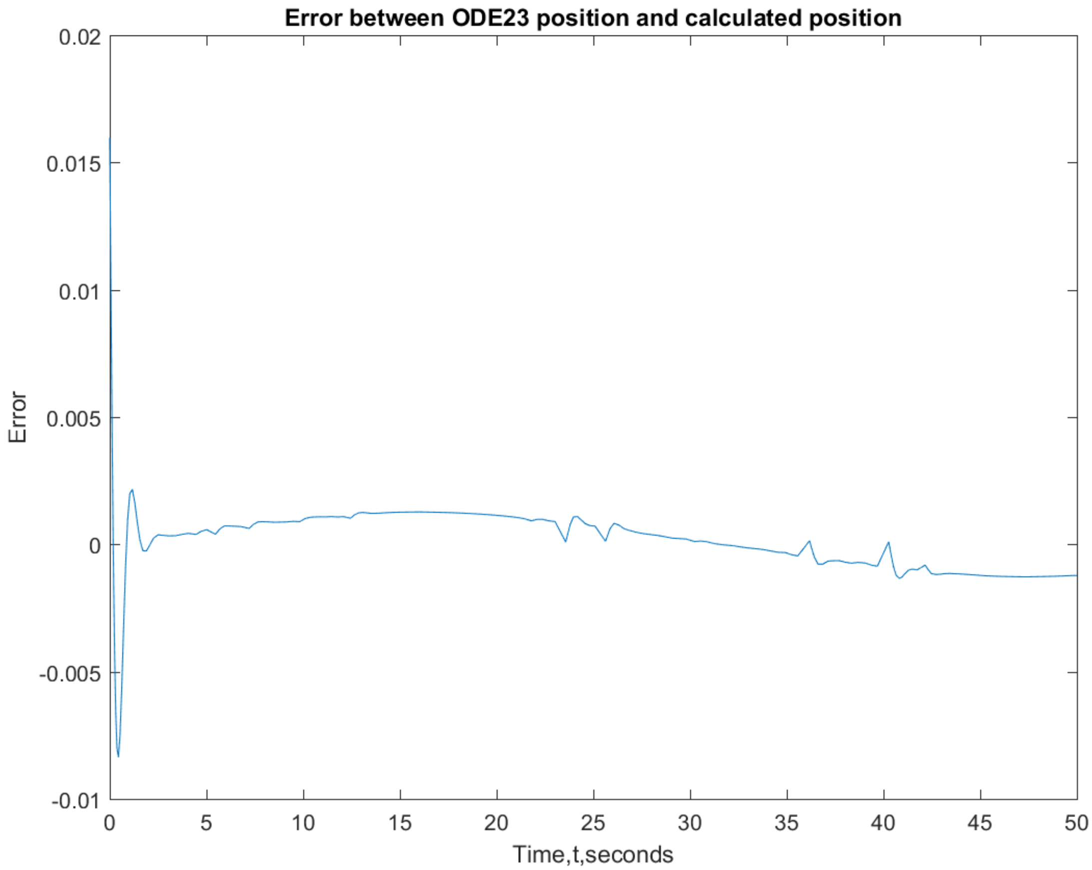

Figure 15: The difference between ODE23 and the expected value.

The error values are extremely small, being no more then 2% away from the predicted value.

Appendix

```
function [ y,t,x] = getLinSimResults( t_end,ts,u,x0 )
%getLinSimResults Runs a linear simulation of the TF given in Lab 2
%   This function runs a linear simualation with the transfer function
%   25/(s^2+4s+25). The simulation runs from 0 to t_end and returns the
%   results.

% Time Vector
t = 0:ts:t_end;

% State space model.
A = [0 1;-25 -4];
B = [0;25];
C = [1 0]; % Both are set to one here so we can get the state
D = 0;

% Linear Simulation
[y,x]=lsim(A,B,C,D,u,t,x0);

```

```
end
```

The above function was used as a helper function that would allow me to rapidly get the simulation results for a given situation.

```
function [ dx ] = fn( t,x )
%fn same as f, but does not need u, will calc u from t
%
dx = [0 1;-25 -4]*x + [0;25]*sin(0.1*t);

end
```

The above function was used to generate the ODE23 plots with the input function of $u(t) = sin(0.1t)$

Table Values with Sinusoidal input

Table 1: Euler Simulations for different h

	h=5.0			h=1.0			h=0.1			h=0.01		
Time, t	k	x_1	x_2	k	x_1	x_2	k	x_1	x_2	k	x_1	x_2
0.0	0	3	0	0	3	0	0	3	0	0	3	0
5.0	1	3	-375	5	$-0.5e^4$	$-1.8e^4$	50	0.52	-0.13	500	0.47	0.09
10.0	2	-1872	6810	10	$0.3e^7$	$7.1e^7$	100	0.83	0.05	1000	0.83	0.06
15.0	3	32180	104720	15	$0.1e^{11}$	$-1.9e^{11}$	150	1.0	0.01	1500	1.0	0.01
20.0	4	$0.6e^6$	$-6.0e^6$	20	$-0.6e^{14}$	$4.0e^{14}$	200	0.92	-0.04	2000	0.92	-0.04
25.0	5	$-3.0e^7$	$-4.5e^7$	25	$1.9e^{17}$	$-6.1e^{17}$	250	0.61	-0.08	2500	0.61	-0.08
30.0	6	$0.2e^9$	$2.8e^9$	30	$-4.4e^{20}$	$3.8e^{20}$	300	0.16	-0.1	3000	0.16	-0.1
35.0	7	$1.4e^{10}$	$-7.8e^{10}$	35	$0.8e^{24}$	$1.6e^{24}$	350	-0.34	-0.09	3500	-0.34	-0.09
40.0	5	$-3.8e^{11}$	$-3.1e^{11}$	40	$-0.8e^{27}$	$-8.5e^{27}$	400	-0.75	-0.07	4000	-0.75	-0.07
45.0	9	$-0.2e^{13}$	$5.3e^{13}$	45	$-0.1e^{31}$	$2.5e^{31}$	450	-0.97	-0.02	4500	-0.97	-0.02
50.0	10	$2.6e^{14}$	$-7.6e^{14}$	50	$0.7e^{34}$	$-5.8e^{34}$	500	-0.96	0.02	5000	-0.96	0.03
CPUTIME		0.0034 s			0.004 s			0.006 s			0.02 s	

From here we can clearly see that as h increases so does the amount of time required to simulate. But as h increase we can see that the simulation also gets more accurate.

Whole Lab 2 code

```
clear all;
close all;

x0=[3;0];
T=0.01;
N = round(50/T)+1;
t = zeros(1,N);
x = zeros(2,N);
x(:,1) = x0;
t = T*(0:N-1);
u = 0.*t;
%u = sin(0.1.*t);
tic;

for i=1:N
   dx = f(t(i),x(:,i),u(i));
   x(:,i+1) = x(:,i) + dx*T;
```

```
end
toc

clf;
y=x(1,1:i);
y2=x(2,1:i);

subplot(2,1,1);
plot(t,y);
str = sprintf('Euler Simulation (x1) for T = %g',T);
title(str);
xlabel('Time,t,seconds');
ylabel('Position,x1');

subplot(2,1,2);
plot(t,y2);
str = sprintf('Euler Simulation (x2) for T = %g',T);
title(str);
xlabel('Time,t,seconds');
ylabel('Velocity,x2');


tstr = sprintf('t_%g',T);
tstr = strrep(tstr, '.', '_');
filename = sprintf('es_%s',tstr);
print(filename,'-dpng');

%% Linsim
[y_l,time,x] = getLinSimResults(50,T,u,x0);

y1_l=x(:,1);
y2_l=x(:,2);

figure;
subplot(2,1,1)
plot(t,y1_l);
str = sprintf('Linsim (x1) result for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Position,y');

subplot(2,1,2)
plot(t,y2_l);
str = sprintf('Linsim (x2) result for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Velocity,y2');


filename = sprintf('ls_%s',tstr);
print(filename,'-dpng');


%% Error Calc
error = y-y1_l';
error2 = y2-y2_l';

figure;
subplot(2,1,1)
```

```
plot(t,error)
str = sprintf('Error between Eulers simulation and linsim (x1) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Error');

subplot(2,1,2)
plot(t,error2)
str = sprintf('Error between Eulers simulation and linsim (x2) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Error');


filename = sprintf('e_%s',tstr);
print(filename,'-dpng');

%% ODE23
tic
[t_ode,y_ode]=ode23(@(t_ode,y_ode) f(t_ode,y_ode,0),[0 50],[3;0]);
toc
figure;
subplot(2,1,1)
plot(t_ode,y_ode(:,1));
str = sprintf('ODE23(x1) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Position,x1');

subplot(2,1,2)
plot(t_ode,y_ode(:,2));
str = sprintf('ODE23(x2) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Velocity,x2');

tstr = sprintf('t_%g',T);
tstr = strrep(tstr, '.', '_');
filename = sprintf('ode23');
print(filename,'-dpng');

[t_ode2,y_ode2]=ode23(@(t_ode2,y_ode2) fn(t_ode2,y_ode2),[0 50],[3;0]);
figure;
subplot(2,1,1)
plot(t_ode2,y_ode2(:,1));
str = sprintf('ODE23(x1) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Position,x1');

subplot(2,1,2)
plot(t_ode2,y_ode2(:,2));
str = sprintf('ODE23(x2) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Velocity,x2');

filename = sprintf('ode23_sin_wi');
print(filename,'-dpng');
```

```
[t_ode3,y_ode3]=ode23(@(t_ode2,y_ode3) fn(t_ode2,y_ode3),[0 50],[0;0]);
figure;
subplot(2,1,1)
plot(t_ode3,y_ode3(:,1));
str = sprintf('ODE23(x1) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Position,x1');

subplot(2,1,2)
plot(t_ode3,y_ode3(:,2));
str = sprintf('ODE23(x2) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Velocity,x2');

filename = sprintf('ode23_sin_ni');
print(filename,'-dpng');

%% The two plots on top of one another between ODE and eulers
figure;
subplot(2,1,1)
plot(t_ode,y_ode(:,1),t,y);
str = sprintf('ODE23 and Eulers Simulation (x1) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Position,x1');
legend('ODE23 Result','Eulers Simulation');

subplot(2,1,2)
plot(t_ode,y_ode(:,2),t,2);
str = sprintf('ODE23 and Eulers Simulation (x2) for T = %g',T);
title(str)
xlabel('Time,t,seconds');
ylabel('Velocity,x2');
legend('ODE23 Result','Eulers Simulation');

filename = sprintf('ode23_es');
print(filename,'-dpng');

%% ODE error from prediction:
y_predicted = 0.999*sin(t_ode3*0.1 - deg2rad(0.9163));
error3 = y_ode3(:,1) - y_predicted;
figure;
plot(t_ode3,error3);
title('Error between ODE23 position and calculated position');
ylabel('Error')
xlabel('Time,t,seconds');

filename = sprintf('ode23_error');
print(filename,'-dpng');
```