

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:

$$\omega_n = 5$$

$$G = 1$$

$$\zeta = \frac{2}{5}$$

$$\sigma = \zeta\omega_n = 2$$

$$\omega_d = \omega_n\sqrt{1 - \zeta^2} = 3.8729$$

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:

$$y(t) = 0.999\sin(0.1t - 0.9163^\circ)$$

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

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

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

$$\begin{aligned}
A &= \begin{bmatrix} 0 & 1 \\ -25 & -4 \end{bmatrix} & B &= \begin{bmatrix} 0 \\ 25 \end{bmatrix} \\
C &= [1 \quad 0] & D &= [0]
\end{aligned}$$

Lab Results

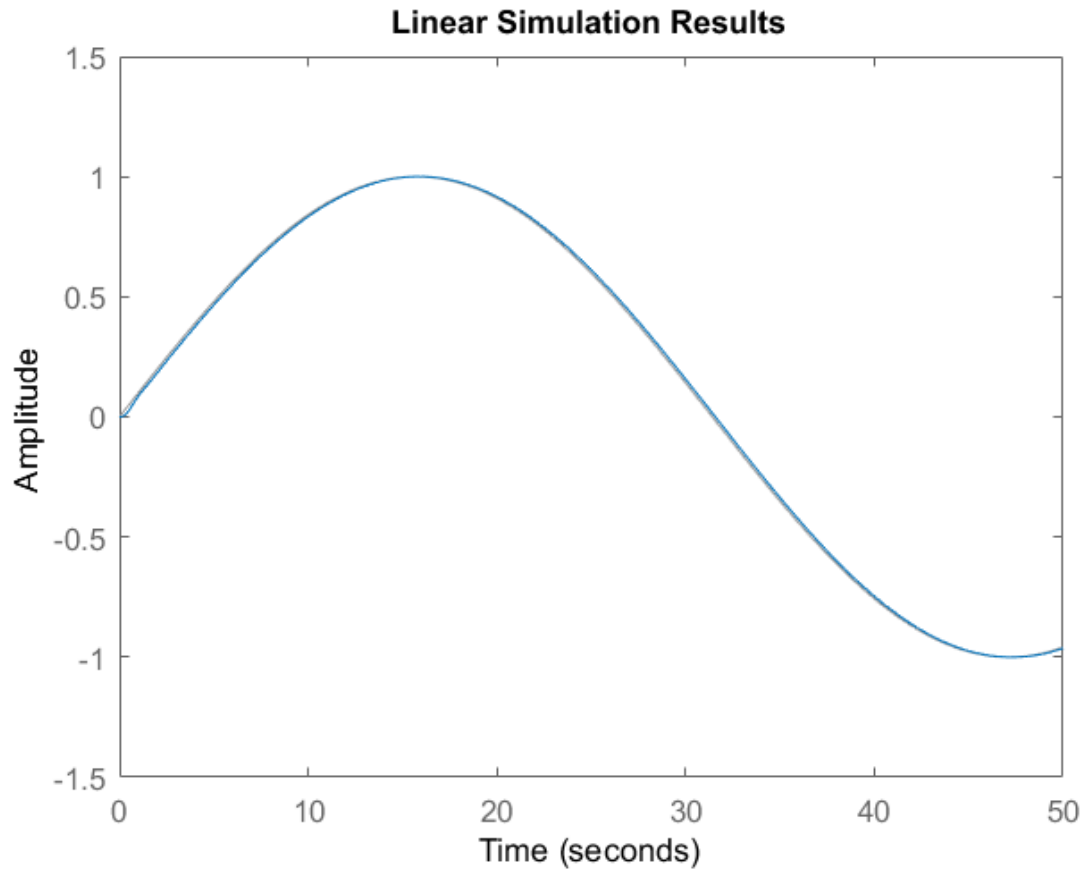


Figure 1: Results of lsim using $u(t) = \sin(0.1t)$

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.

```
1 function [ dx ] = f( t,x,u )
2 %f This function does the one step of the simulation
3
4 dx = [ 0 1;-25 -4]*x + [0;25]*u;
5
6 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.

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

```

7  t = T*(0:N-1);
8  u = 0.*t;
9  %u = sin(0.1.*t);
10 %flops(0);
11
12 for i=1:N
13     dx = f(t(i),x(:,i),u(i));
14     x(:,i+1) = x(:,i) + dx*T;
15 end
16 %flops
17
18 clf;
19 y=x(1,1:i);
20 y2=x(2,1:i);
21
22 subplot(2,1,1);
23 plot(t,y);
24 str = sprintf('Euler Simulation (x1) for T = %g',T);
25 title(str);
26 xlabel('Time,t,seconds');
27 ylabel('Position,x1');
28
29 subplot(2,1,2);
30 plot(t,y2);
31 str = sprintf('Euler Simulation (x2) for T = %g',T);
32 title(str);
33 xlabel('Time,t,seconds');
34 ylabel('Velocity,x2');
35
36
37 tstr = sprintf('t_%g',T);
38 tstr = strrep(tstr, '.', '_');
39 filename = sprintf('es_%s',tstr);
40 print(filename,'-dpng');
41
42 %% Linsim
43 [y_1,time,x] = getLinSimResults(50,T,u,x0);
44
45 y1_l=x(:,1);
46 y2_l=x(:,2);
47
48 figure;
49 subplot(2,1,1)
50 plot(t,y1_l);
51 str = sprintf('Linsim (x1) result for T = %g',T);
52 title(str)
53 xlabel('Time,t,seconds');
54 ylabel('Position,y');
55
56 subplot(2,1,2)
57 plot(t,y2_l);
58 str = sprintf('Linsim (x2) result for T = %g',T);
59 title(str)
60 xlabel('Time,t,seconds');
61 ylabel('Velocity,y2');
62
63
64 filename = sprintf('ls_%s',tstr);
65 print(filename,'-dpng');

```

```

66
67
68 %% Error Calc
69 error = y-y1_1';
70 error2 = y2-y2_1';
71
72 figure;
73 subplot(2,1,1)
74 plot(t,error)
75 str = sprintf('Error between Eulers simulation and linsim (x1) for T = %g',T);
76 title(str)
77 xlabel('Time,t,seconds');
78 ylabel('Error');
79
80 subplot(2,1,2)
81 plot(t,error2)
82 str = sprintf('Error between Eulers simulation and linsim (x2) for T = %g',T);
83 title(str)
84 xlabel('Time,t,seconds');
85 ylabel('Error');
86
87
88 filename = sprintf('e_%s',tstr);
89 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)

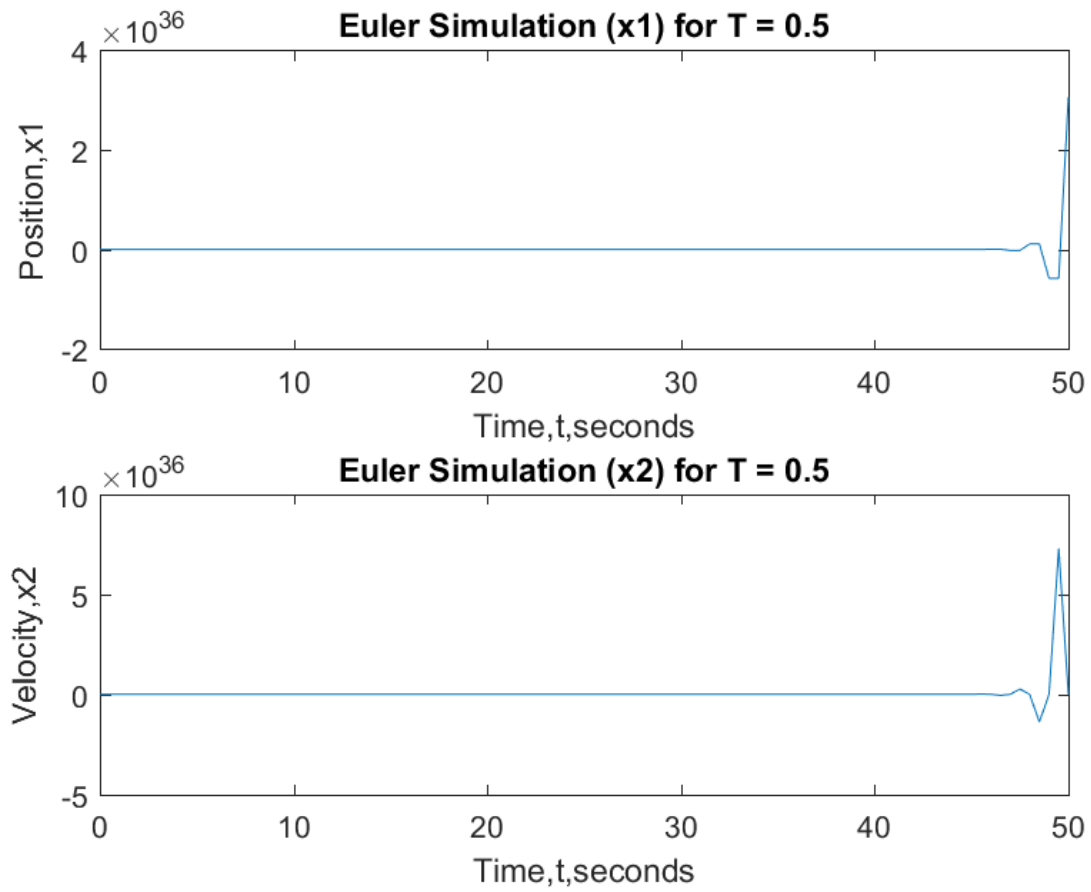


Figure 2: The results of the Euler simulation with just a initial condition and no input

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

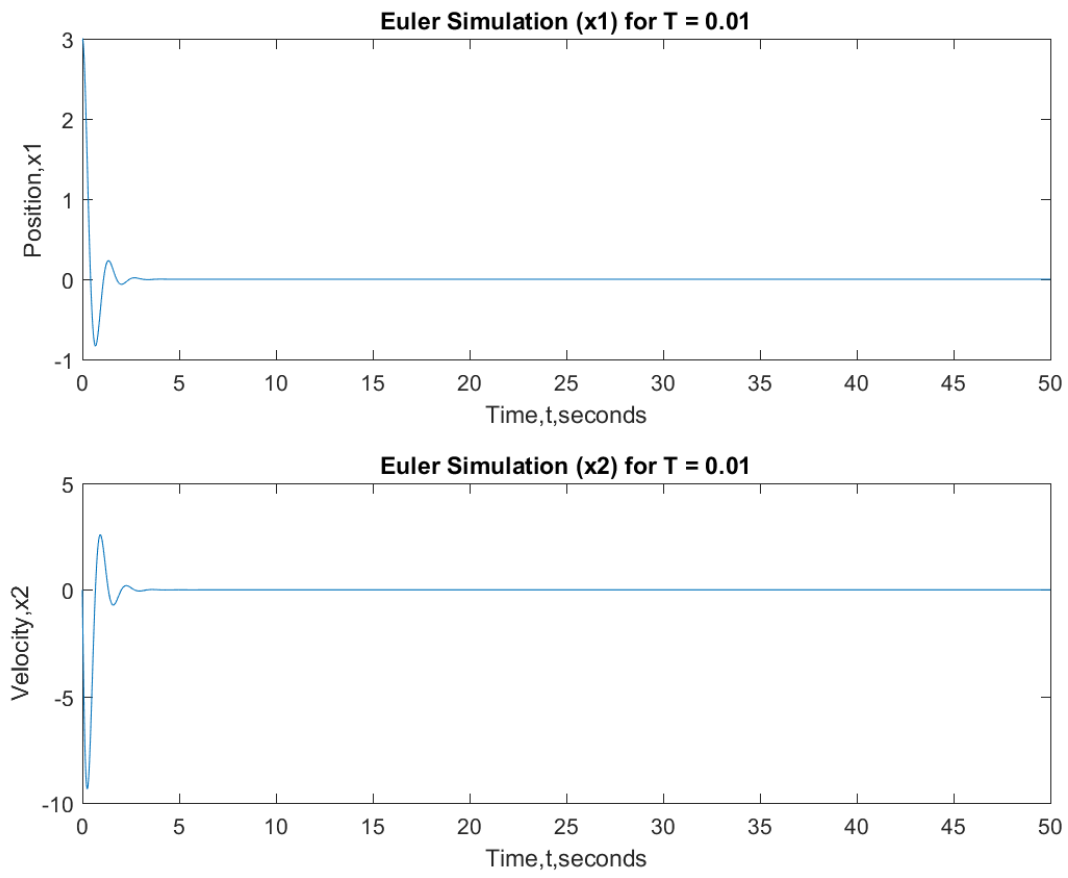


Figure 3: The results of the Euler simulation with just a initial condition and no input

This simulation took 0.02 seconds.

No Input and Initial Conditions = 0

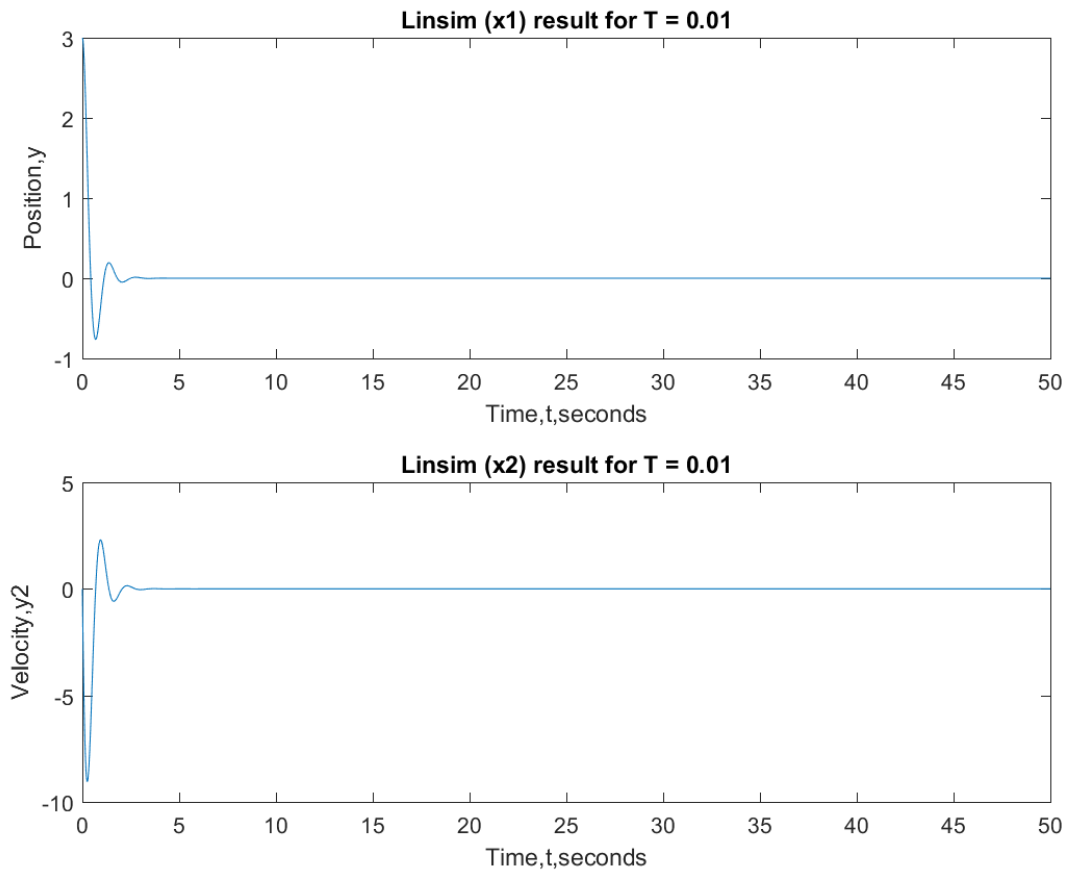


Figure 4: The results of the Linear simulation with just a initial condition and no input

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

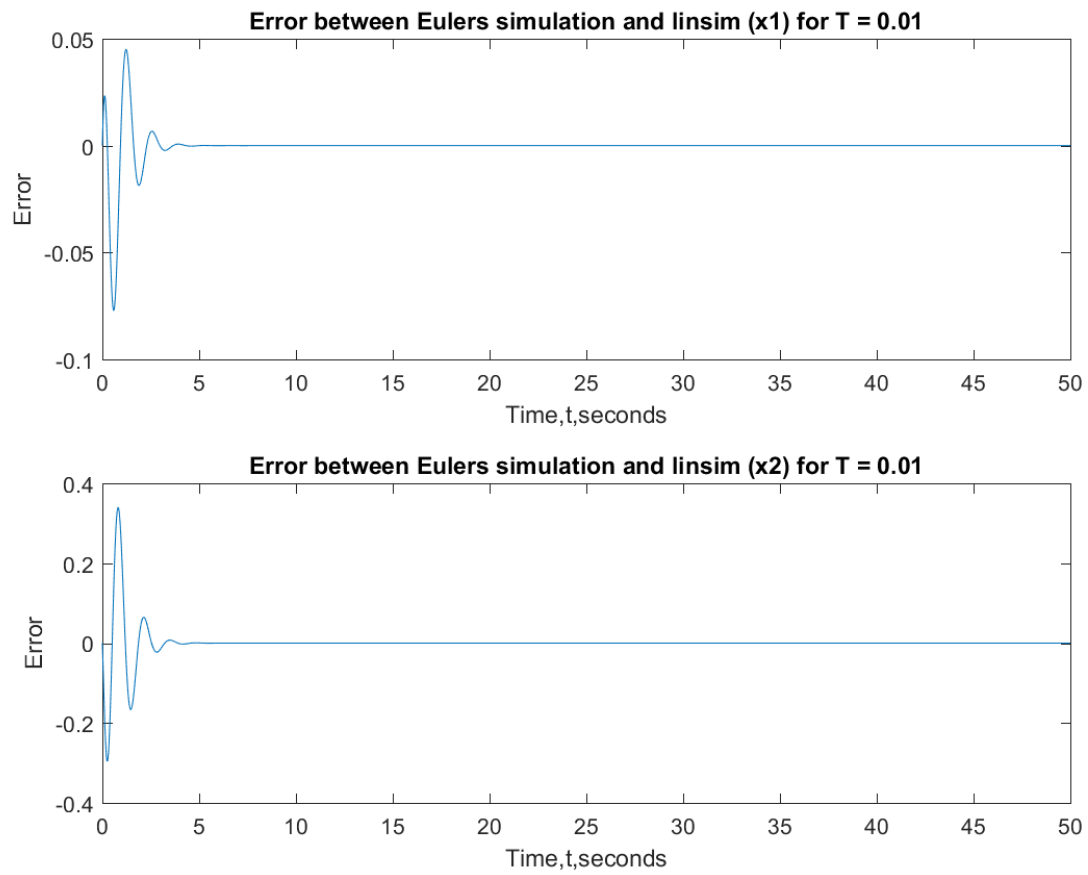


Figure 5: The results of the Euler simulation with just a initial condition and no input

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

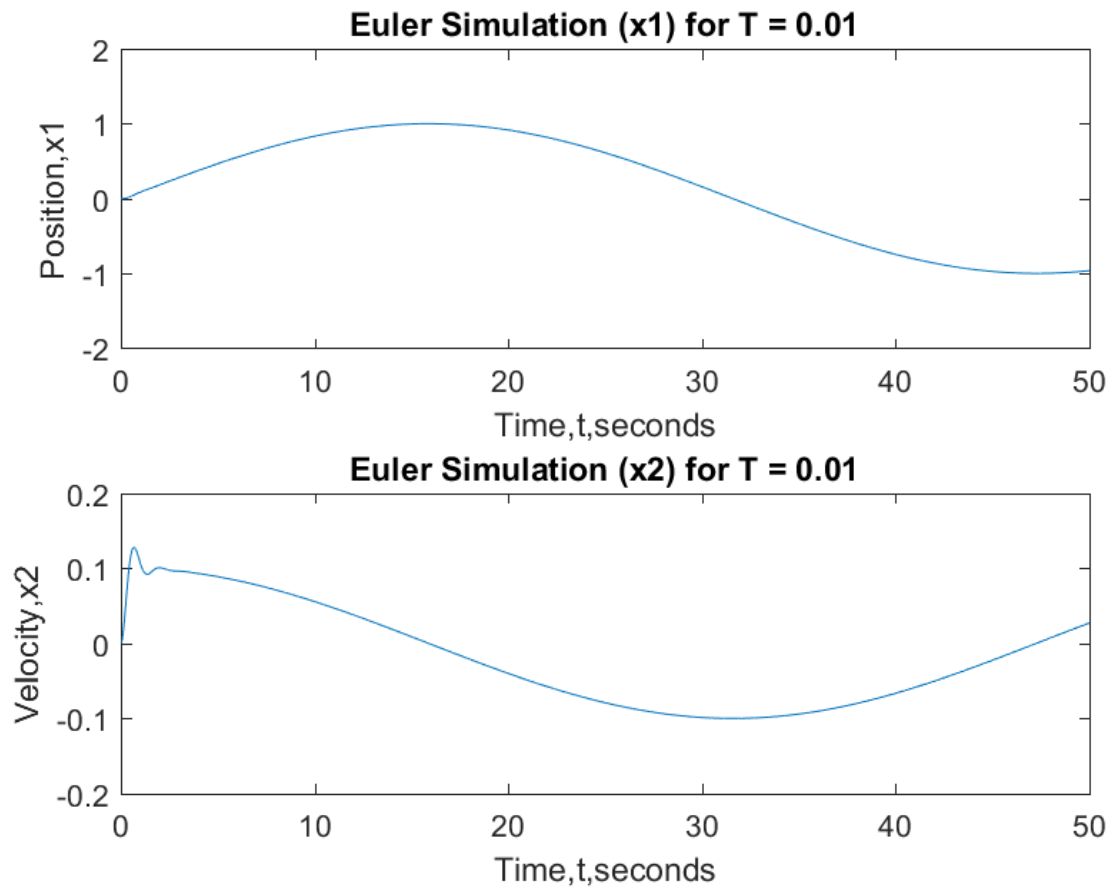


Figure 6: The results of the Euler simulation

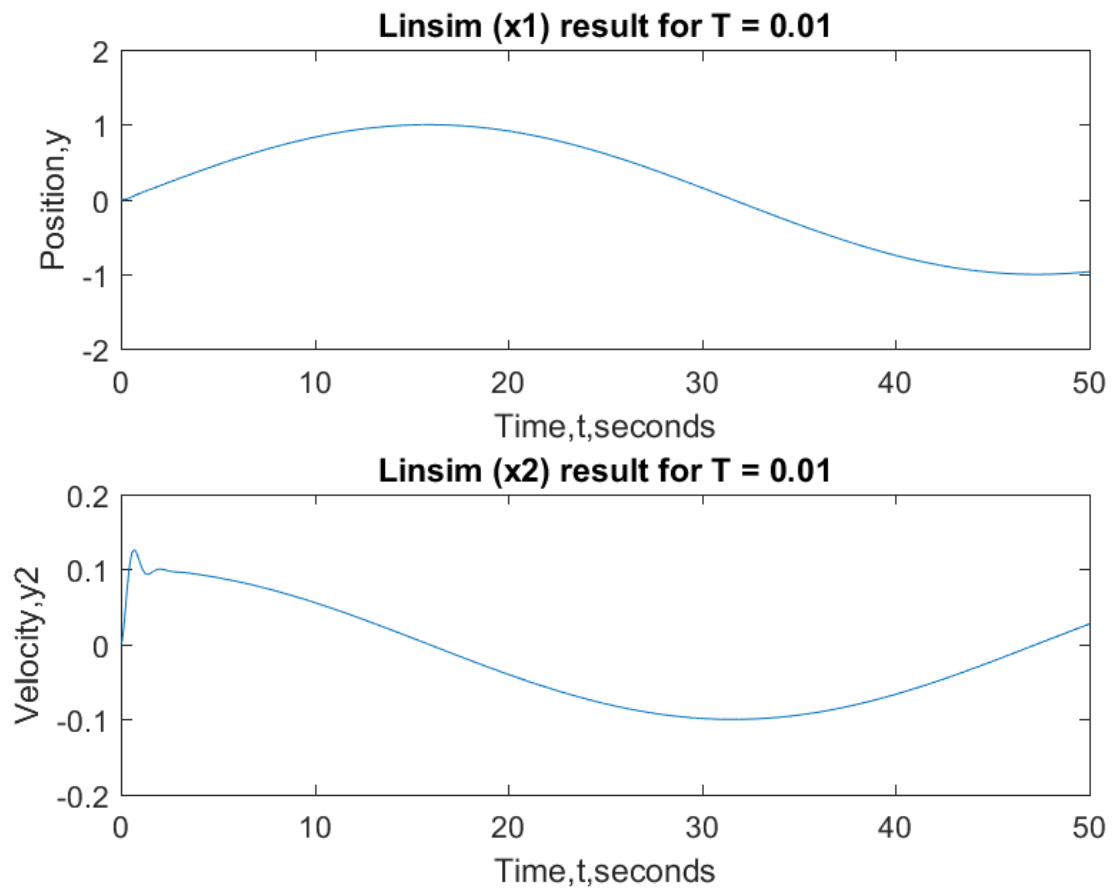


Figure 7: The results of the linear simulation

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

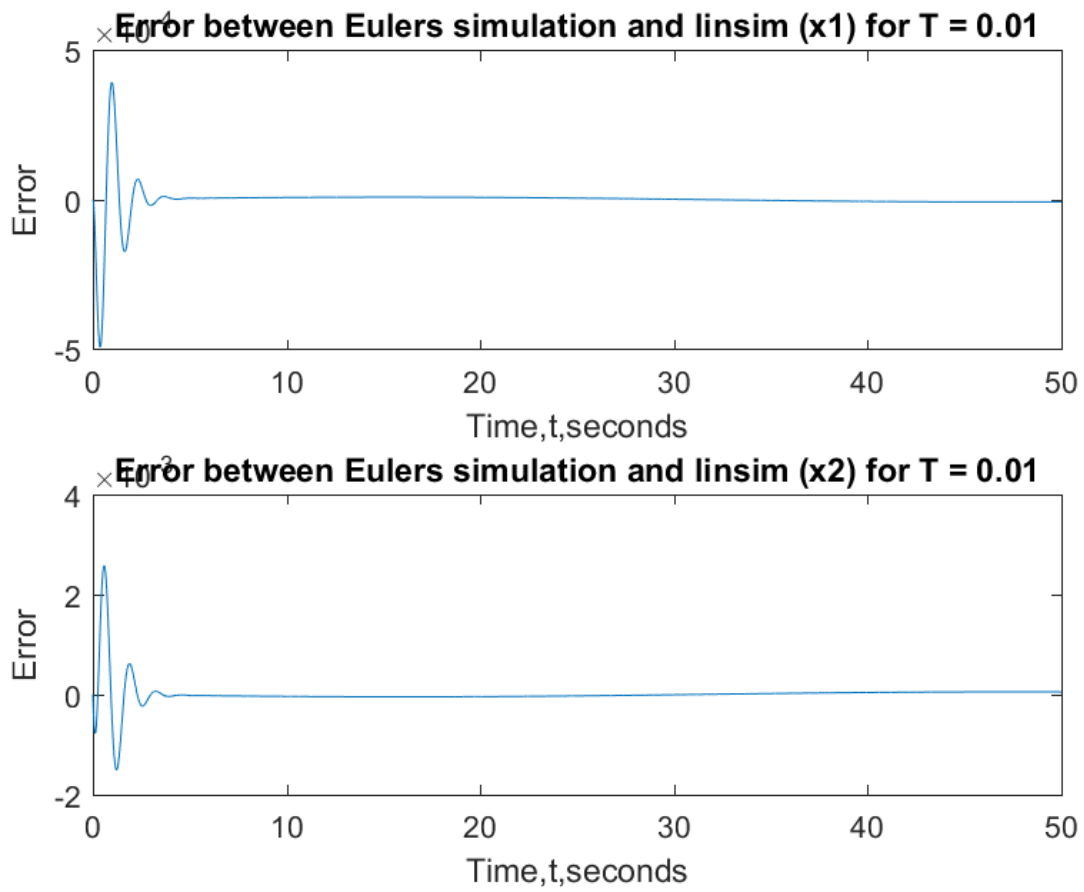


Figure 8: The difference between euler simulation and linsim

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 = $[3;0]$

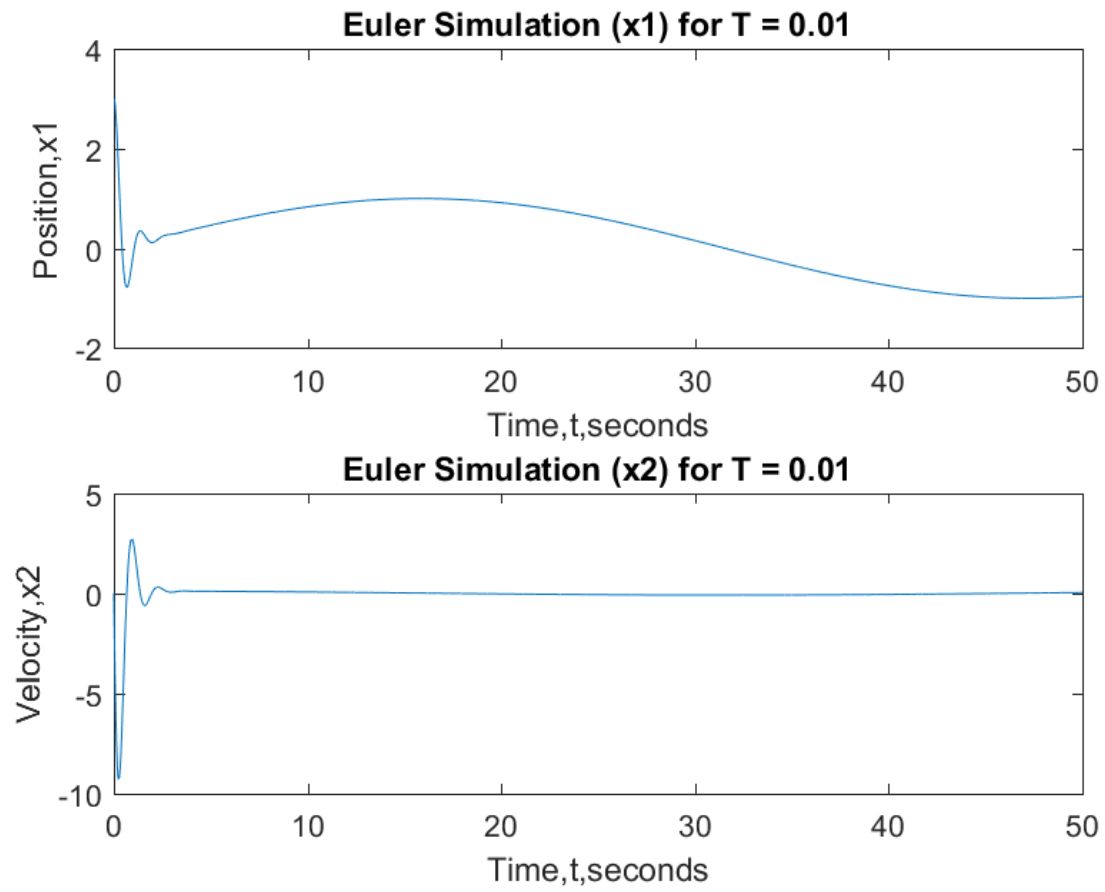


Figure 9: The results of the Euler simulation

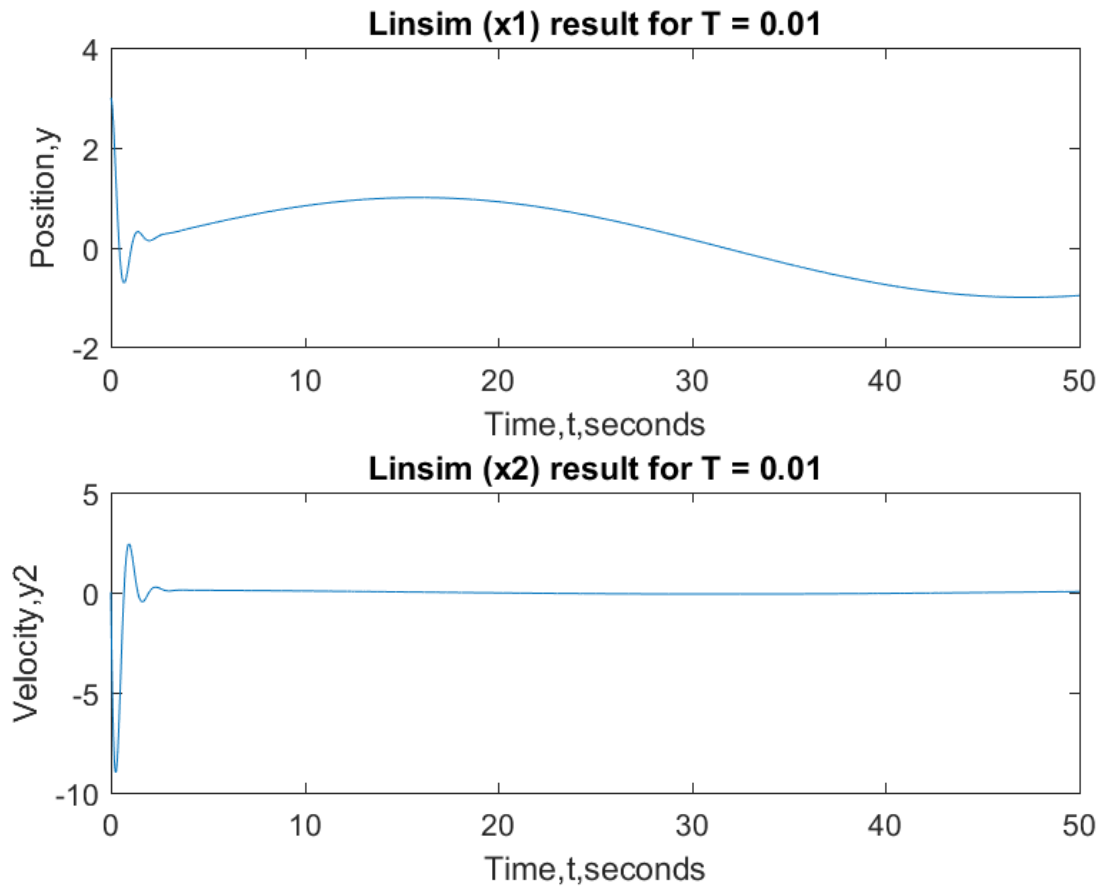


Figure 10: The results of the linear simulation

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

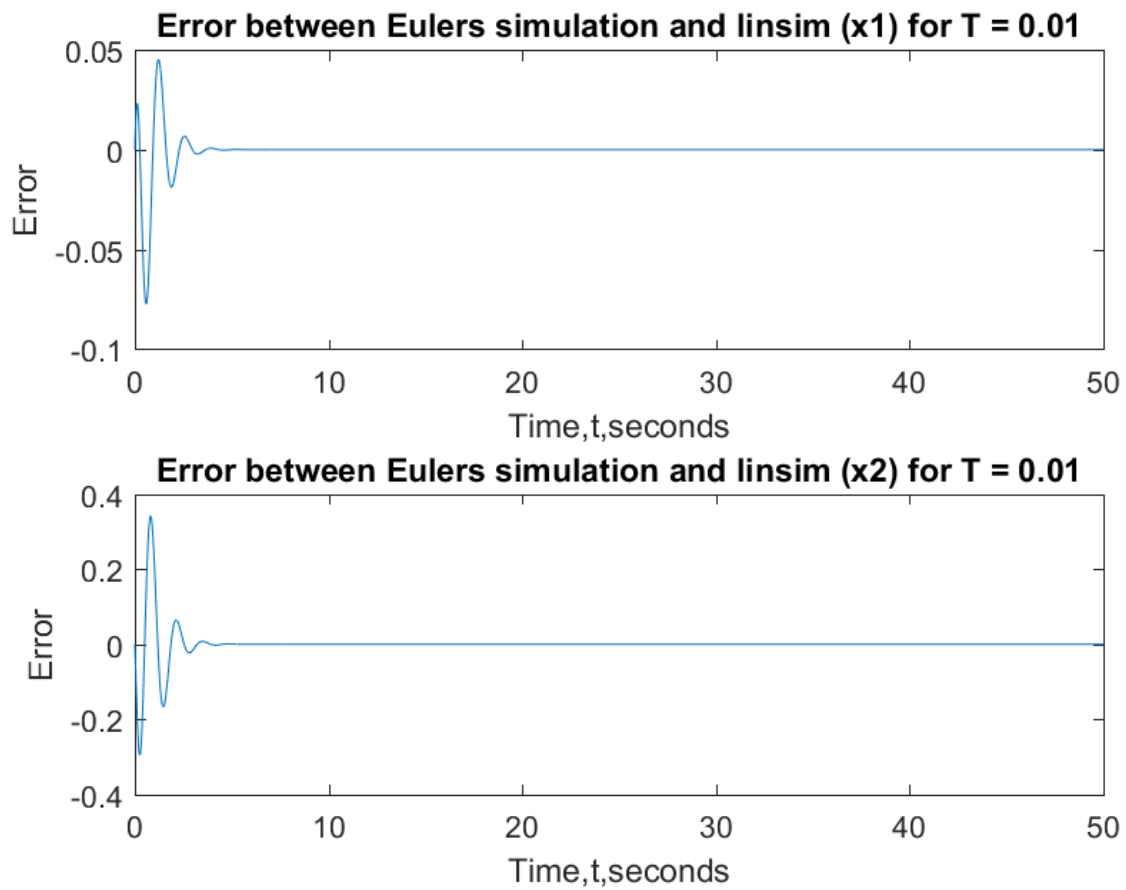


Figure 11: The error between the euler simulation and linsim

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.

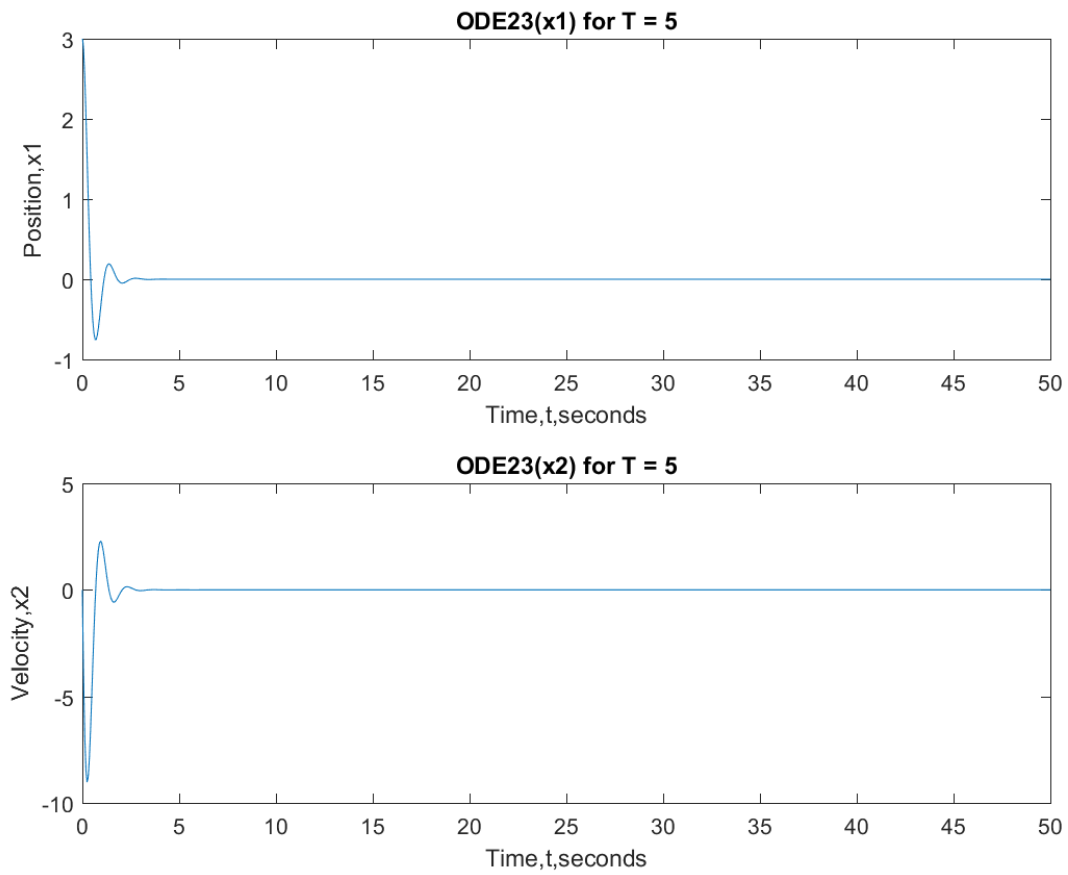


Figure 12: The results of the ODE23 with just a initial condition and no input

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

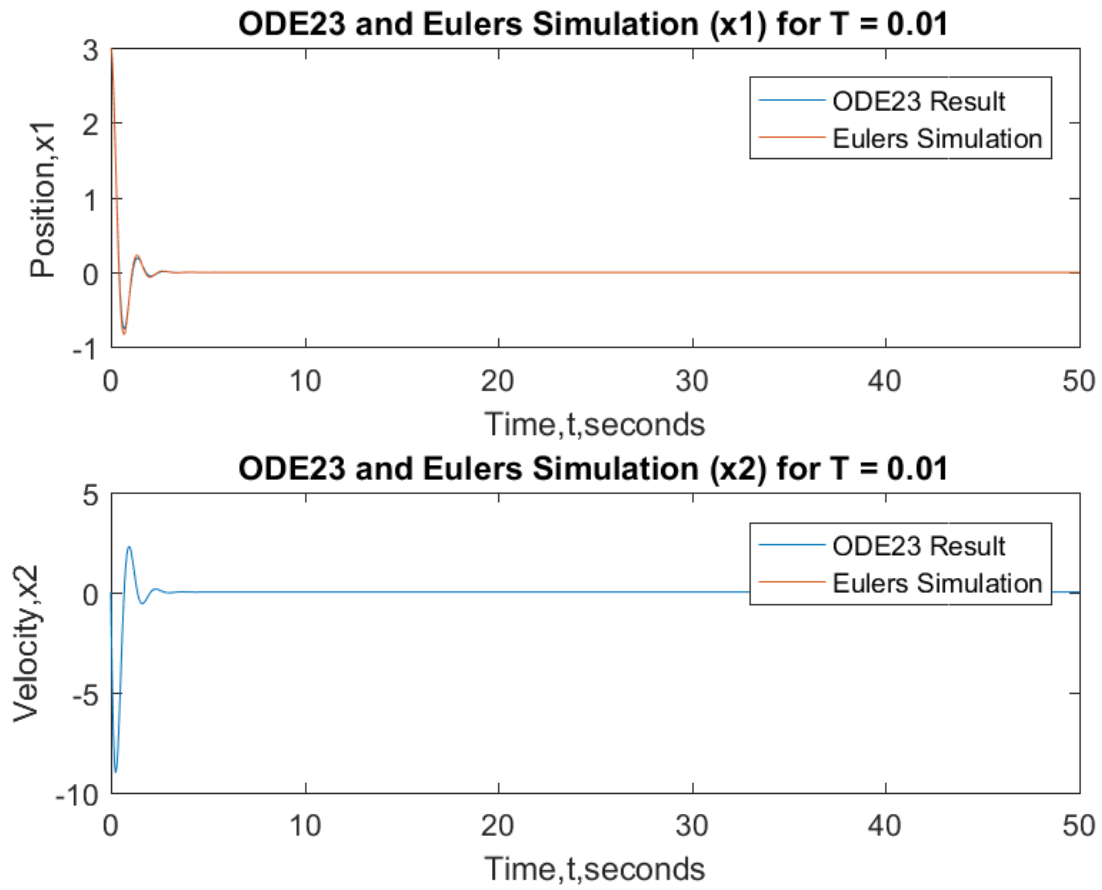


Figure 13: The results of the ODE23 overlaid with the Euler's Simulation.

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:

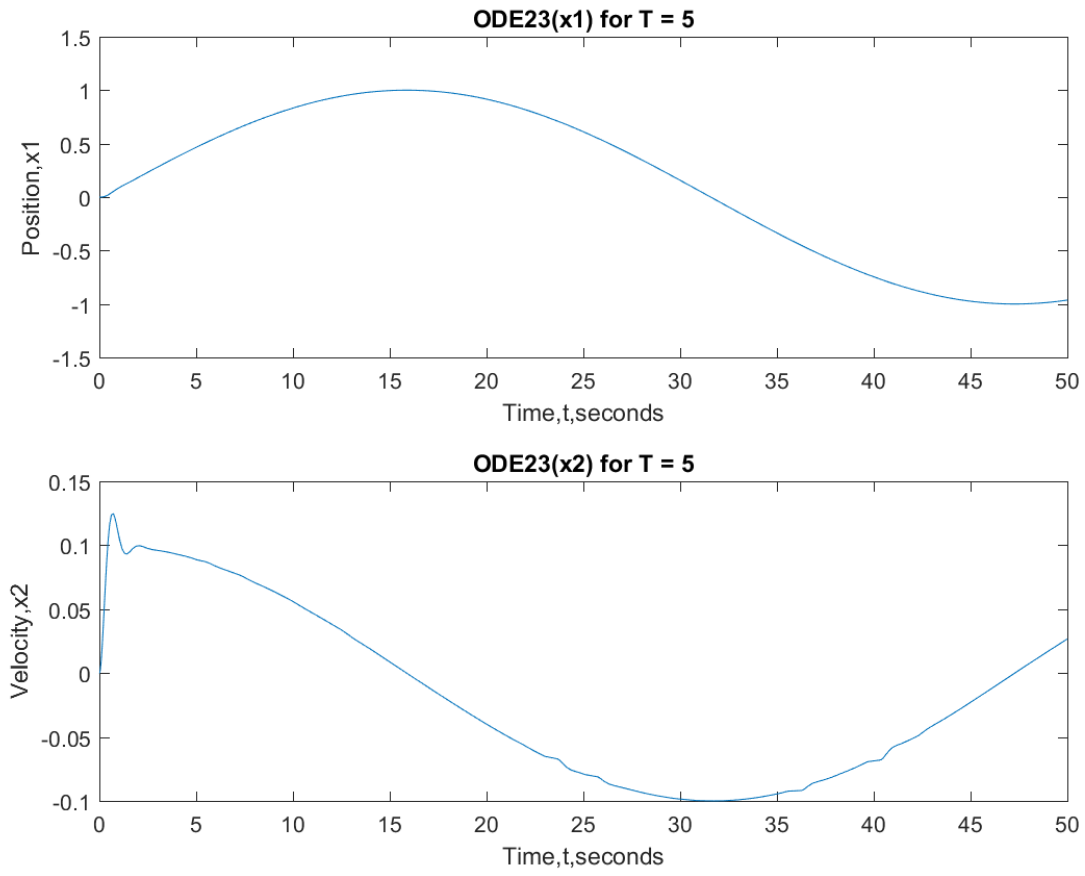


Figure 14: The results of the ODE23 simulation with an input of $u(t) = \sin(0.1t)$

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

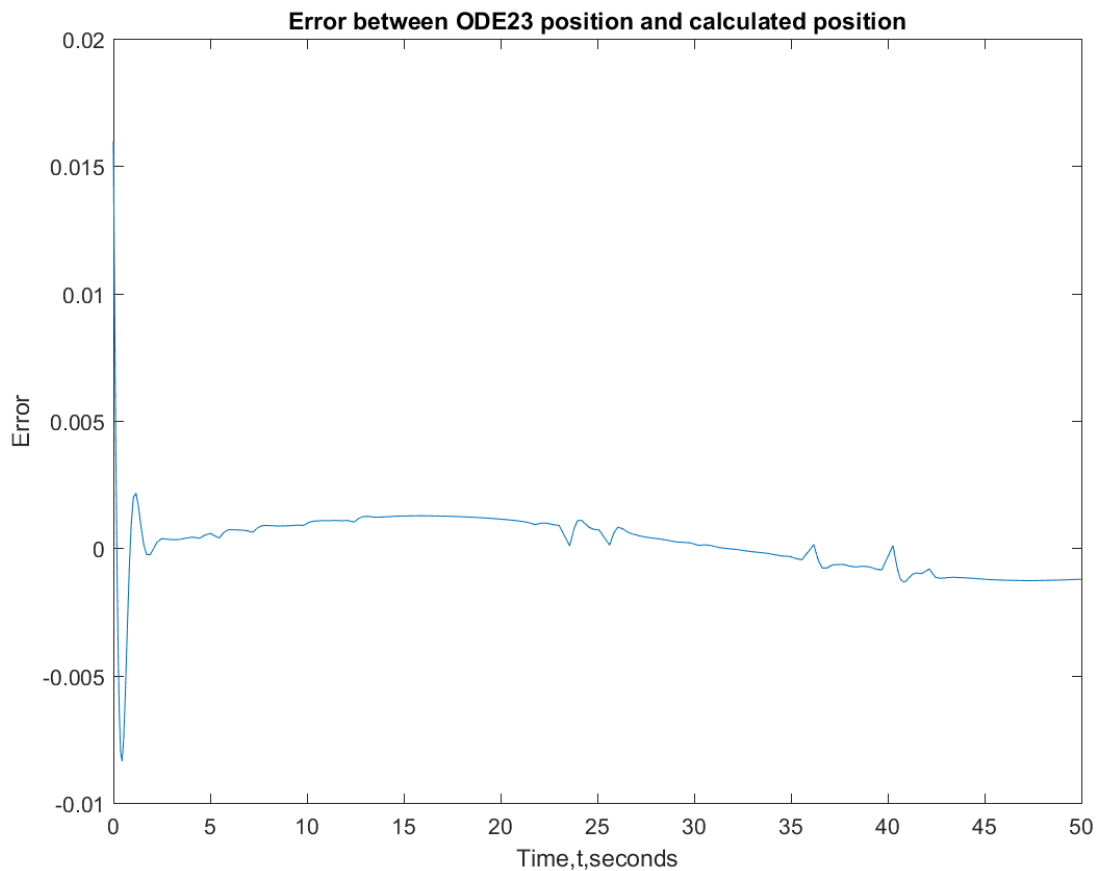


Figure 15: The difference between ODE23 and the expected value.

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

Appendix

```

1 function [ y,t,x] = getLinSimResults( t_end,ts,u,x0 )
2 %getLinSimResults Runs a linear simulation of the TF given in Lab 2
3 % This function runs a linear simulation with the transfer function
4 % 25/(s^2+4s+25). The simulation runs from 0 to t_end and returns the
5 % results.
6
7 % Time Vector
8 t = 0:ts:t_end;
9
10 % State space model.
11 A = [0 1;-25 -4];
12 B = [0;25];
13 C = [1 0]; % Both are set to one here so we can get the state
14 D = 0;
15
16 % Linear Simulation
17 [y,x]=lsim(A,B,C,D,u,t,x0);
18

```

19 `end`

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

```
1 function [ dx ] = fn( t,x )
2 %fn same as f, but does not need u, will calc u from t
3 %
4 dx = [0 1;-25 -4]*x + [0;25]*sin(0.1*t);
5
6 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

```
1 clear all;
2 close all;
3
4 x0=[3;0];
5 T=0.01;
6 N = round(50/T)+1;
7 t = zeros(1,N);
8 x = zeros(2,N);
9 x(:,1) = x0;
10 t = T*(0:N-1);
11 u = 0.*t;
12 %u = sin(0.1.*t);
13 tic;
14
15 for i=1:N
16     dx = f(t(i),x(:,i),u(i));
17     x(:,i+1) = x(:,i) + dx*T;
```

```

18 end
19 toc
20
21 clf;
22 y=x(1,1:i);
23 y2=x(2,1:i);
24
25 subplot(2,1,1);
26 plot(t,y);
27 str = sprintf('Euler Simulation (x1) for T = %g',T);
28 title(str);
29 xlabel('Time,t,seconds');
30 ylabel('Position,x1');
31
32 subplot(2,1,2);
33 plot(t,y2);
34 str = sprintf('Euler Simulation (x2) for T = %g',T);
35 title(str);
36 xlabel('Time,t,seconds');
37 ylabel('Velocity,x2');
38
39
40 tstr = sprintf('t_%g',T);
41 tstr = strrep(tstr, '.', '_');
42 filename = sprintf('es_%s',tstr);
43 print(filename,'-dpng');
44
45 %% Linsim
46 [y_1,time,x] = getLinSimResults(50,T,u,x0);
47
48 y1_1=x(:,1);
49 y2_1=x(:,2);
50
51 figure;
52 subplot(2,1,1)
53 plot(t,y1_1);
54 str = sprintf('Linsim (x1) result for T = %g',T);
55 title(str)
56 xlabel('Time,t,seconds');
57 ylabel('Position,y');
58
59 subplot(2,1,2)
60 plot(t,y2_1);
61 str = sprintf('Linsim (x2) result for T = %g',T);
62 title(str)
63 xlabel('Time,t,seconds');
64 ylabel('Velocity,y2');
65
66
67 filename = sprintf('ls_%s',tstr);
68 print(filename,'-dpng');
69
70
71 %% Error Calc
72 error = y-y1_1';
73 error2 = y2-y2_1';
74
75 figure;
76 subplot(2,1,1)

```

```

77 plot(t,error)
78 str = sprintf('Error between Eulers simulation and linsim (x1) for T = %g',T);
79 title(str)
80 xlabel('Time,t,seconds');
81 ylabel('Error');
82
83 subplot(2,1,2)
84 plot(t,error2)
85 str = sprintf('Error between Eulers simulation and linsim (x2) for T = %g',T);
86 title(str)
87 xlabel('Time,t,seconds');
88 ylabel('Error');
89
90
91 filename = sprintf('e_%s',tstr);
92 print(filename,'-dpng');
93
94 %% ODE23
95 tic
96 [t_ode,y_ode]=ode23(@(t_ode,y_ode) f(t_ode,y_ode,0),[0 50],[3;0]);
97 toc
98 figure;
99 subplot(2,1,1)
100 plot(t_ode,y_ode(:,1));
101 str = sprintf('ODE23(x1) for T = %g',T);
102 title(str)
103 xlabel('Time,t,seconds');
104 ylabel('Position,x1');
105
106 subplot(2,1,2)
107 plot(t_ode,y_ode(:,2));
108 str = sprintf('ODE23(x2) for T = %g',T);
109 title(str)
110 xlabel('Time,t,seconds');
111 ylabel('Velocity,x2');
112
113 tstr = sprintf('t_%g',T);
114 tstr = strrep(tstr, '.', '_');
115 filename = sprintf('ode23');
116 print(filename,'-dpng');
117
118 [t_ode2,y_ode2]=ode23(@(t_ode2,y_ode2) fn(t_ode2,y_ode2),[0 50],[3;0]);
119 figure;
120 subplot(2,1,1)
121 plot(t_ode2,y_ode2(:,1));
122 str = sprintf('ODE23(x1) for T = %g',T);
123 title(str)
124 xlabel('Time,t,seconds');
125 ylabel('Position,x1');
126
127 subplot(2,1,2)
128 plot(t_ode2,y_ode2(:,2));
129 str = sprintf('ODE23(x2) for T = %g',T);
130 title(str)
131 xlabel('Time,t,seconds');
132 ylabel('Velocity,x2');
133
134 filename = sprintf('ode23_sin_wi');
135 print(filename,'-dpng');

```

```

136
137 [t_ode3,y_ode3]=ode23(@(t_ode2,y_ode3) fn(t_ode2,y_ode3),[0 50],[0;0]);
138 figure;
139 subplot(2,1,1)
140 plot(t_ode3,y_ode3(:,1));
141 str = sprintf('ODE23(x1) for T = %g',T);
142 title(str)
143 xlabel('Time,t,seconds');
144 ylabel('Position,x1');
145
146 subplot(2,1,2)
147 plot(t_ode3,y_ode3(:,2));
148 str = sprintf('ODE23(x2) for T = %g',T);
149 title(str)
150 xlabel('Time,t,seconds');
151 ylabel('Velocity,x2');
152
153 filename = sprintf('ode23_sin_ni');
154 print(filename,'-dpng');
155
156 %% The two plots on top of one another between ODE and eulers
157 figure;
158 subplot(2,1,1)
159 plot(t_ode,y_ode(:,1),t,y);
160 str = sprintf('ODE23 and Eulers Simulation (x1) for T = %g',T);
161 title(str)
162 xlabel('Time,t,seconds');
163 ylabel('Position,x1');
164 legend('ODE23 Result','Eulers Simulation');
165
166 subplot(2,1,2)
167 plot(t_ode,y_ode(:,2),t,2);
168 str = sprintf('ODE23 and Eulers Simulation (x2) for T = %g',T);
169 title(str)
170 xlabel('Time,t,seconds');
171 ylabel('Velocity,x2');
172 legend('ODE23 Result','Eulers Simulation');
173
174 filename = sprintf('ode23_es');
175 print(filename,'-dpng');
176
177 %% ODE error from prediction:
178 y_predicted = 0.999*sin(t_ode3*0.1 - deg2rad(0.9163));
179 error3 = y_ode3(:,1) - y_predicted;
180 figure;
181 plot(t_ode3,error3);
182 title('Error between ODE23 position and calculated position');
183 ylabel('Error')
184 xlabel('Time,t,seconds');
185
186 filename = sprintf('ode23_error');
187 print(filename,'-dpng');

```
