

ELEC 442 101

Introduction to Robotics

Assignment 5

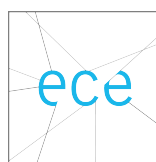
Zhi Chuen Tan Hubert Shim

65408361

98504806



THE UNIVERSITY
OF BRITISH COLUMBIA
Applied Science



Electrical and
Computer
Engineering

1 Two-Link Manipulator Open-Loop Simulation

The Euler-Lagrange approach results in a generalized manipulator dynamics equation of the form:

$$\underbrace{D(\mathbf{q})}_{\text{manipulator inertia matrix}} \ddot{\mathbf{q}} + \underbrace{C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}}_{\text{accounts for Coriolis \& Centripetal terms}} + \underbrace{G(\mathbf{q})}_{\text{accounts for gravitational \& other potential energy terms}} = \underbrace{\mathbf{u}}_{\text{generalized motor forces}} + \underbrace{\underline{J}_n^T \begin{bmatrix} \mathbf{f}_e \\ \boldsymbol{\tau}_e \end{bmatrix}}_{\text{forces from environment}}$$

As the robot does not interact with the environment, we can ignore the last term. Rearranging this equation then gives us:

$$\ddot{\mathbf{q}} = D^{-1}(\mathbf{q})[\mathbf{u} - C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - G(\mathbf{q})], \quad (1)$$

where

$$\mathbf{q} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$\mathbf{u} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

$$D(\mathbf{q}) = \begin{bmatrix} (m_1 + m_2)l_1^2 + m_2l_2^2 + 2m_2l_1l_2 \cos \theta_2 & m_2l_2^2 + m_2l_1l_2 \cos \theta_2 \\ m_2l_2^2 + m_2l_1l_2 \cos \theta_2 & m_2l_2^2 \end{bmatrix}$$

$$C(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -2m_2l_1l_2(\sin \theta_2)\dot{\theta}_2 & -m_2l_1l_2(\sin \theta_2)\dot{\theta}_2 \\ m_2l_1l_2(\sin \theta_2)\dot{\theta}_1 & 0 \end{bmatrix}$$

$$G(\mathbf{q}) = \begin{bmatrix} (m_1 + m_2)gl_1 \cos \theta_1 + m_2gl_2 \cos (\theta_1 + \theta_2) \\ m_2gl_2 \cos (\theta_1 + \theta_2) \end{bmatrix}$$

$$T = \frac{1}{2} \dot{\mathbf{q}}^T D(\mathbf{q}) \dot{\mathbf{q}}$$

$$V = m_1gl_1 \sin \theta_1 + m_2g[l_1 \sin \theta_1 + l_2 \sin (\theta_1 + \theta_2)]$$

$$\Rightarrow E_T = T + V$$

The simulation results for each run can be found in section 1.1.3.

1.1 Simulation

1.1.1 Simulink block diagrams

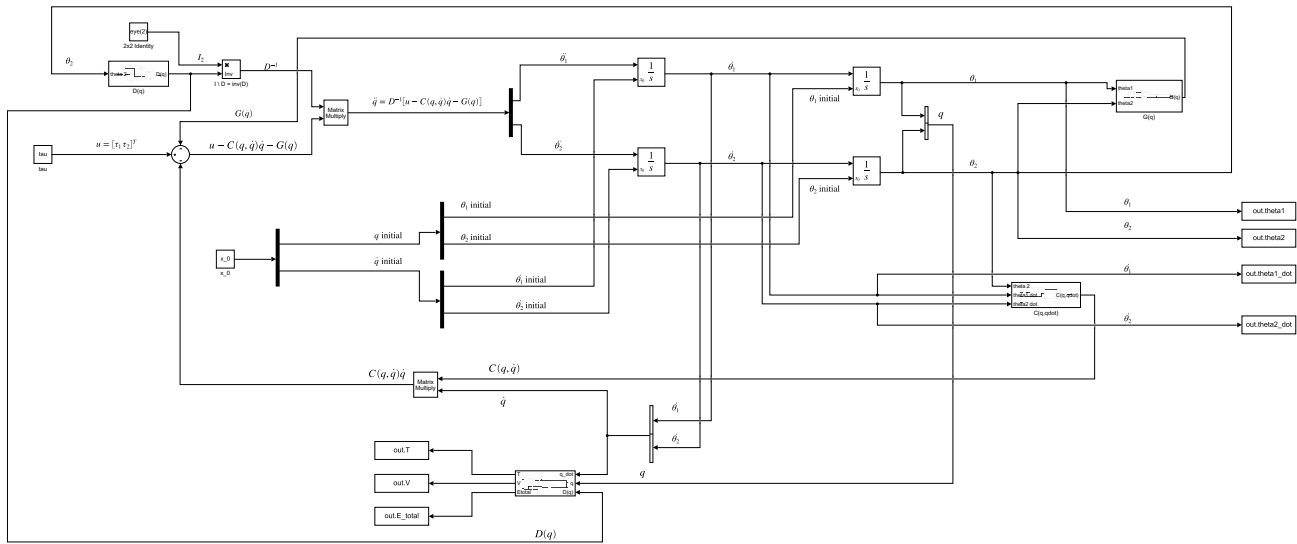


Figure 1: Simulink Block Diagram for the robot.

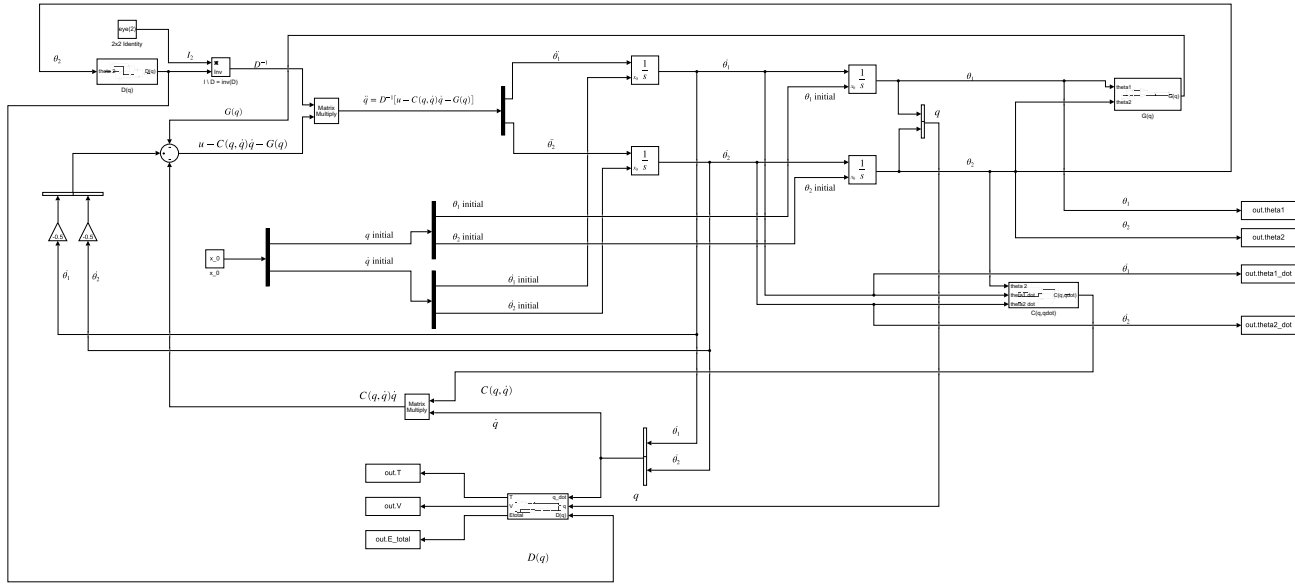


Figure 2: Simulink Block Diagram for the robot, used in part 1(c) to feed friction back into the system.)

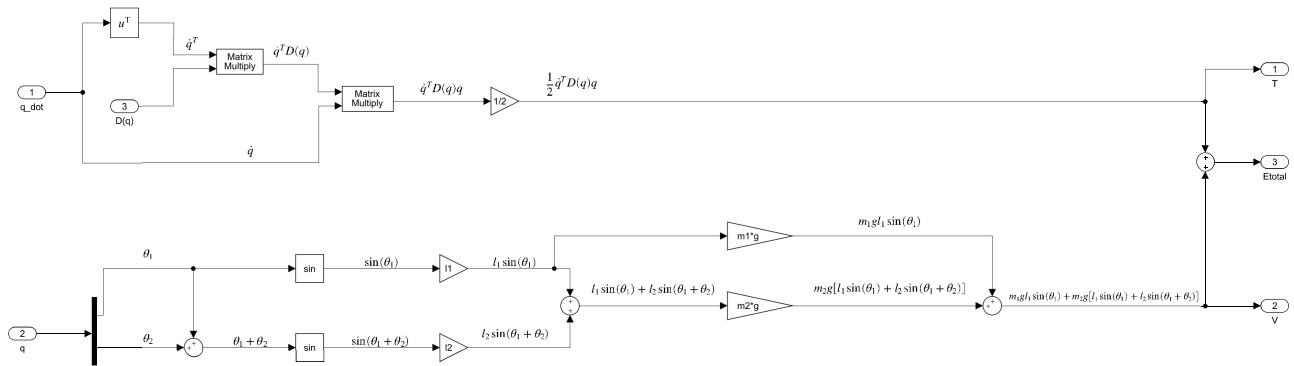


Figure 3: Simulink Block Diagram for the the subsystem used to calculate (potetial, kinetic, and total) energy.

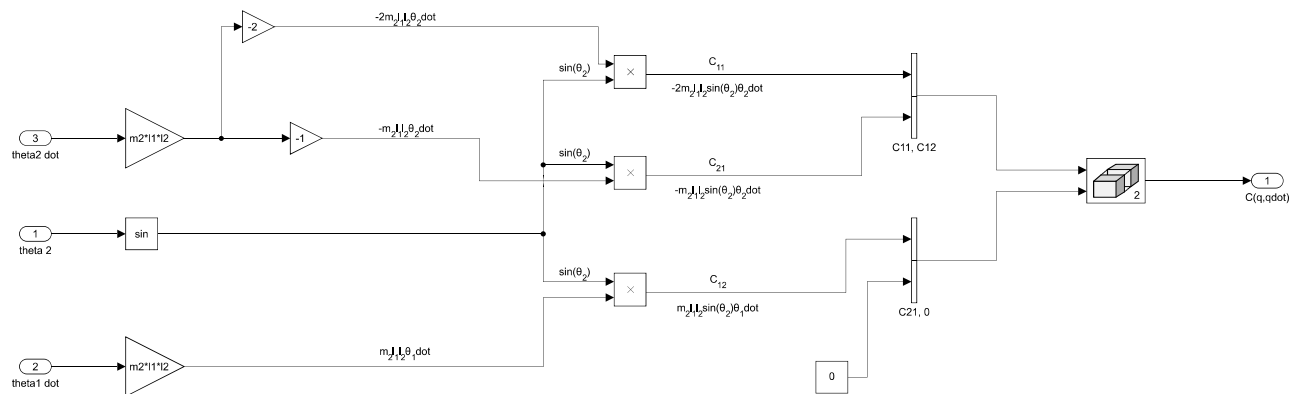


Figure 5: Simulink Block Diagram for the the subsystem used to calculate the $C(q, \dot{q})$ matrix.

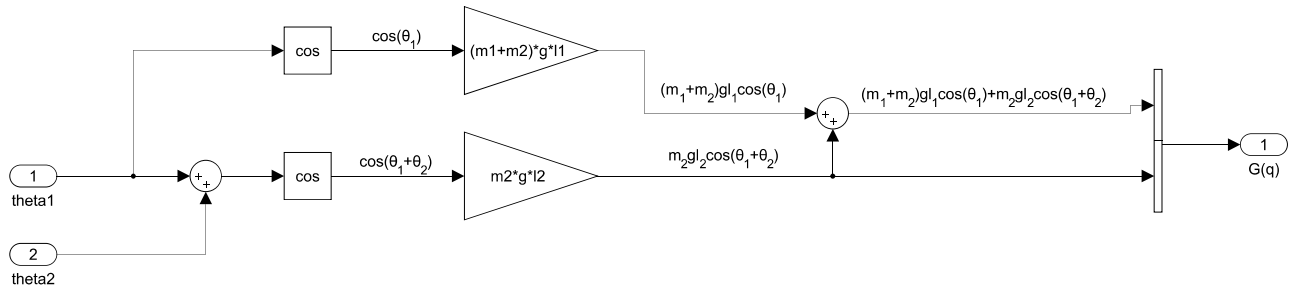


Figure 6: Simulink Block Diagram for the the subsystem used to calculate the $G(q)$ matrix.

1.1.2 MATLAB code

The following code was used to simplify running the simulation, plotting the joint angles and velocities, and kinetic, potential, and total energies, and automatically saves the figures as `.fig` (and `.eps` or `.pdf` files for the \LaTeX typesetting, which makes these diagrams vector diagrams, i.e. they can be zoomed-in without significant aliasing):

```
1 function run_sim_asn5q1(x_0_in,tau_in)
2     % Initialize constants
3     x_0 = x_0_in;
4     tau = tau_in;
5     l1=1;l2=1;m1=1;m2=1;g=9.81;
```



```
6
7     % Run simulation
8     simOut = sim('asn5q1');
9
10    % Take output values from Simulink
11    theta1=simOut.get('theta1');
12    theta2=simOut.get('theta2');
13    theta1_dot=simOut.get('theta1_dot');
14    theta2_dot=simOut.get('theta2_dot');
15    T=simOut.get('T');
16    V=simOut.get('V');
17    Eltotal=simOut.get('Eltotal');
18
19    % Plots theta1
20    figure;
21    hold on;
22    view(2);
23    title('Plot for  $\theta_1$ ', 'Interpreter', 'latex');
24    xlabel('Time (seconds)', 'Interpreter', 'latex');
25    ylabel(' $\theta_1$ ', 'Interpreter', 'latex');
26    plot(theta1, 'Color', '#A2142F');
27    saveas(gcf, 'qlat1.fig'); % saves figure as .fig
28    saveas(gcf, 'qlat1', 'eps'); % saves figure as .eps (for preparing text)
29
30    % Plots theta2
31    figure;
32    hold on;
33    view(2);
34    title('Plot for  $\theta_2$ ', 'Interpreter', 'latex');
35    xlabel('Time (seconds)', 'Interpreter', 'latex');
36    ylabel(' $\theta_2$ ', 'Interpreter', 'latex');
37    plot(theta2, 'Color', '#A2142F');
38    saveas(gcf, 'qlat2.fig'); % saves figure as .fig
39    saveas(gcf, 'qlat2', 'eps'); % saves figure as .eps (for preparing text)
40
41    % Plots theta1_dot
42    figure;
43    hold on;
44    view(2);
45    title('Plot for  $\dot{\theta}_1$ ', 'Interpreter', 'latex');
46    xlabel('Time (seconds)', 'Interpreter', 'latex');
47    ylabel(' $\dot{\theta}_1$ ', 'Interpreter', 'latex');
48    plot(theta1_dot, 'Color', '#A2142F');
49    saveas(gcf, 'qlat1d.fig'); % saves figure as .fig
50    saveas(gcf, 'qlat1d', 'eps'); % saves figure as .eps (for preparing text)
```

```

51
52     % Plots theta2.dot
53     figure;
54     hold on;
55     view(2);
56     title('Plot for  $\dot{\theta}_2$ ', 'Interpreter', 'latex');
57     xlabel('Time (seconds)', 'Interpreter', 'latex');
58     ylabel(' $\dot{\theta}_2$ ', 'Interpreter', 'latex');
59     plot(theta2_dot, 'Color', '#A2142F');
60     saveas(gcf, 'qla_t2d.fig'); % saves figure as .fig
61     saveas(gcf, 'qla_t2d', 'eps'); % saves figure as .eps (for preparing text)
62
63     % Plots kinetic energy
64     figure;
65     hold on;
66     view(2);
67     title('Plot for Kinetic Energy,  $T$ ', 'Interpreter', 'latex');
68     xlabel('Time (seconds)', 'Interpreter', 'latex');
69     ylabel(' $T$ ', 'Interpreter', 'latex');
70     plot(T, 'Color', '#A2142F');
71     saveas(gcf, 'qla_T.fig'); % saves figure as .fig
72     saveas(gcf, 'qla_T', 'eps'); % saves figure as .eps (for preparing text)
73
74     % Plots potential energy
75     figure;
76     hold on;
77     view(2);
78     title('Plot for Potential Energy,  $V$ ', 'Interpreter', 'latex');
79     xlabel('Time (seconds)', 'Interpreter', 'latex');
80     ylabel(' $V$ ', 'Interpreter', 'latex');
81     plot(V, 'Color', '#A2142F');
82     saveas(gcf, 'qla_V.fig'); % saves figure as .fig
83     saveas(gcf, 'qla_V', 'eps'); % saves figure as .eps (for preparing text)
84
85     % Plots total energy
86     figure;
87     hold on;
88     view(2);
89     title('Plot for Total Energy,  $T+V$ ', 'Interpreter', 'latex');
90     xlabel('Time (seconds)', 'Interpreter', 'latex');
91     ylabel(' $T+V$ ', 'Interpreter', 'latex');
92     plot(E_total, 'Color', '#A2142F');
93     saveas(gcf, 'qla_total.fig'); % saves figure as .fig
94     saveas(gcf, 'qla_total', 'eps'); % saves figure as .eps (for preparing text)
95     end

```

Listing 1: MATLAB code used to simulate the system for questions 1(a-b)

```
1 function run_sim_asn5q1c(x_0_in)
2     % Initialize constants
3     x_0 = x_0_in;
4
5     l1=1;l2=1;m1=1;m2=1;g=9.81;
6
7     % Run simulation
8     simOut = sim('asn5q1c');
9
10    % Take output values from Simulink
11    theta1=simOut.get('theta1');
12    theta2=simOut.get('theta2');
13    theta1_dot=simOut.get('theta1_dot');
14    theta2_dot=simOut.get('theta2_dot');
15    T=simOut.get('T');
16    V=simOut.get('V');
17    E_total=simOut.get('E_total');
18
19    % Plots theta1
20    figure;
21    hold on;
22    view(2);
23    title('Plot for  $\theta_1$ ', 'Interpreter', 'latex');
24    xlabel('Time (seconds)', 'Interpreter', 'latex');
25    ylabel(' $\theta_1$ ', 'Interpreter', 'latex');
26    plot(theta1, 'blue');
27    saveas(gcf, 'q1c.t1.fig'); % saves figure as .fig
28    saveas(gcf, 'q1c.t1', 'eps'); % saves figure as .eps (for preparing text)
29
30    % Plots theta2
31    figure;
32    hold on;
33    view(2);
34    title('Plot for  $\theta_2$ ', 'Interpreter', 'latex');
35    xlabel('Time (seconds)', 'Interpreter', 'latex');
36    ylabel(' $\theta_2$ ', 'Interpreter', 'latex');
37    plot(theta2, 'blue');
38    saveas(gcf, 'q1c.t2.fig'); % saves figure as .fig
39    saveas(gcf, 'q1c.t2', 'eps'); % saves figure as .eps (for preparing text)
40
41    % Plots theta1_dot
```

```

42     figure;
43     hold on;
44     view(2);
45     title('Plot for  $\dot{\theta}_1$ ', 'Interpreter', 'latex');
46     xlabel('Time (seconds)', 'Interpreter', 'latex');
47     ylabel(' $\dot{\theta}_1$ ', 'Interpreter', 'latex');
48     plot(theta1_dot, 'blue');
49     saveas(gcf, 'qlc-t1d.fig'); % saves figure as .fig
50     saveas(gcf, 'qlc-t1d', 'eps'); % saves figure as .eps (for preparing text)
51
52     % Plots theta2_dot
53     figure;
54     hold on;
55     view(2);
56     title('Plot for  $\dot{\theta}_2$ ', 'Interpreter', 'latex');
57     xlabel('Time (seconds)', 'Interpreter', 'latex');
58     ylabel(' $\dot{\theta}_2$ ', 'Interpreter', 'latex');
59     plot(theta2_dot, 'blue');
60     saveas(gcf, 'qlc-t2d.fig'); % saves figure as .fig
61     saveas(gcf, 'qlc-t2d', 'eps'); % saves figure as .eps (for preparing text)
62
63     % Plots kinetic energy
64     figure;
65     hold on;
66     view(2);
67     title('Plot for Kinetic Energy,  $T$ ', 'Interpreter', 'latex');
68     xlabel('Time (seconds)', 'Interpreter', 'latex');
69     ylabel(' $T$ ', 'Interpreter', 'latex');
70     plot(T, 'blue');
71     saveas(gcf, 'qlc-T.fig'); % saves figure as .fig
72     saveas(gcf, 'qlc-T', 'eps'); % saves figure as .eps (for preparing text)
73
74     % Plots potential energy
75     figure;
76     hold on;
77     view(2);
78     title('Plot for Potential Energy,  $V$ ', 'Interpreter', 'latex');
79     xlabel('Time (seconds)', 'Interpreter', 'latex');
80     ylabel(' $V$ ', 'Interpreter', 'latex');
81     plot(V, 'blue');
82     saveas(gcf, 'qlc-V.fig'); % saves figure as .fig
83     saveas(gcf, 'qlc-V', 'eps'); % saves figure as .eps (for preparing text)
84
85     % Plots total energy
86     figure;

```

```
87     hold on;
88     view(2);
89     title('Plot for Total Energy,  $T+V$ ', 'Interpreter', 'latex');
90     xlabel('Time (seconds)', 'Interpreter', 'latex');
91     ylabel(' $T+V$ ', 'Interpreter', 'latex');
92     plot(Etotal, 'blue');
93     saveas(gcf, 'qlc-total.fig'); % saves figure as .fig
94     saveas(gcf, 'qlc-total', 'eps'); % saves figure as .eps (for preparing text)
95
96
97
98
99
100 end
```

Listing 2: MATLAB code used to simulate the system for question 1(c)

1.1.3 Simulation results

All of the diagrams below are vector diagrams and can be zoomed in without significant aliasing.

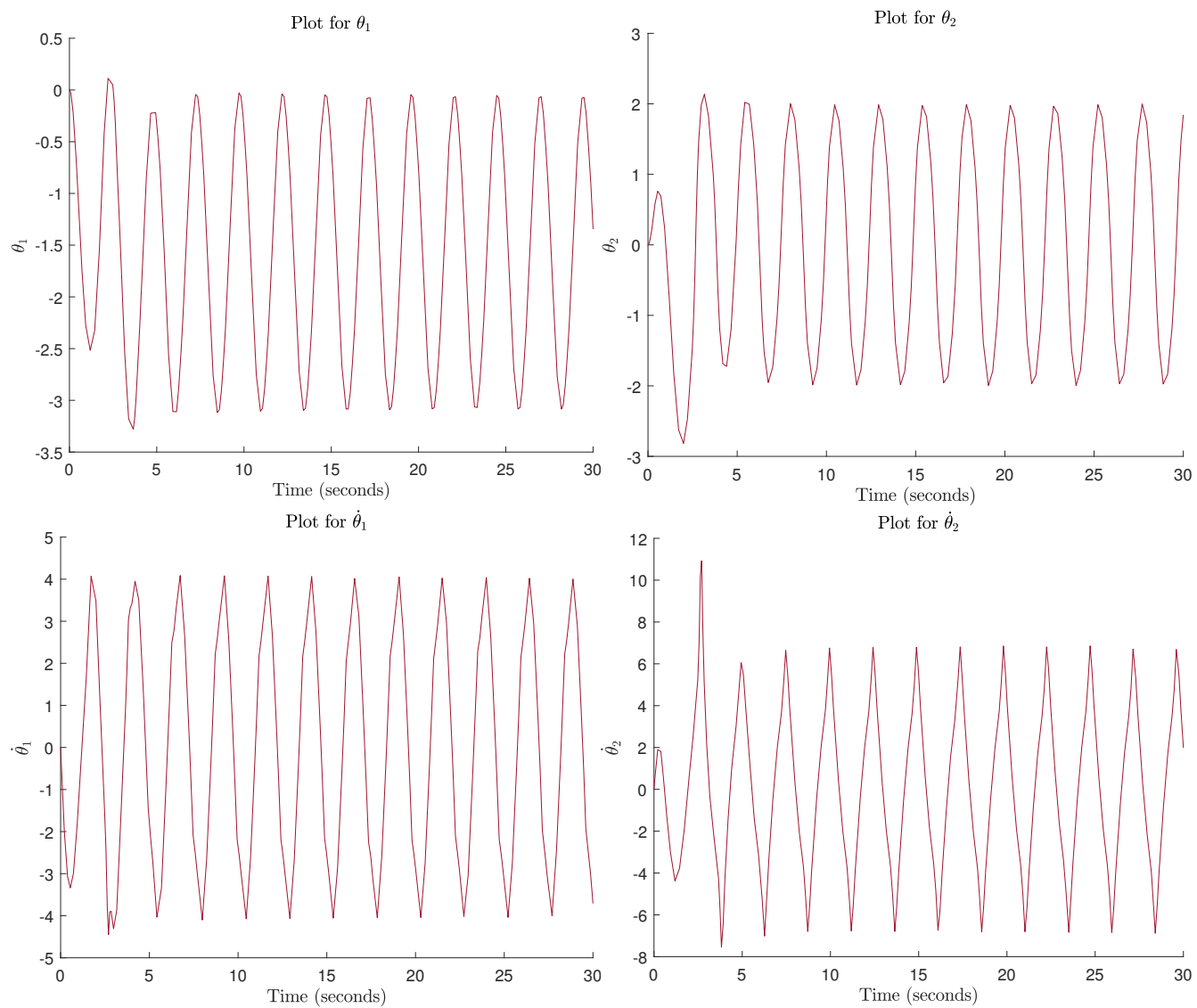


Figure 7: Joint angle and velocity plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = \tau_2 = 0$

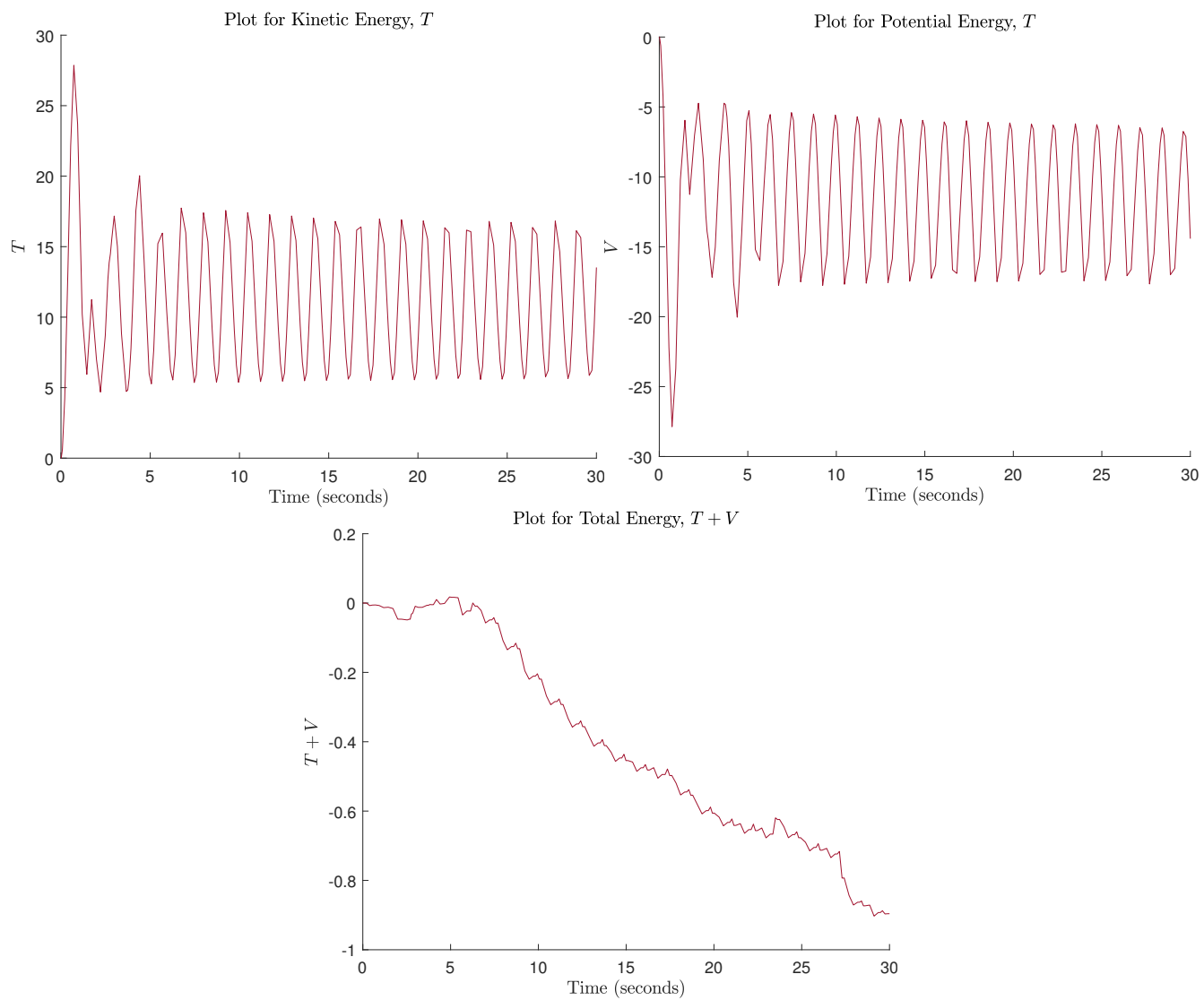


Figure 8: Energy plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = \tau_2 = 0$

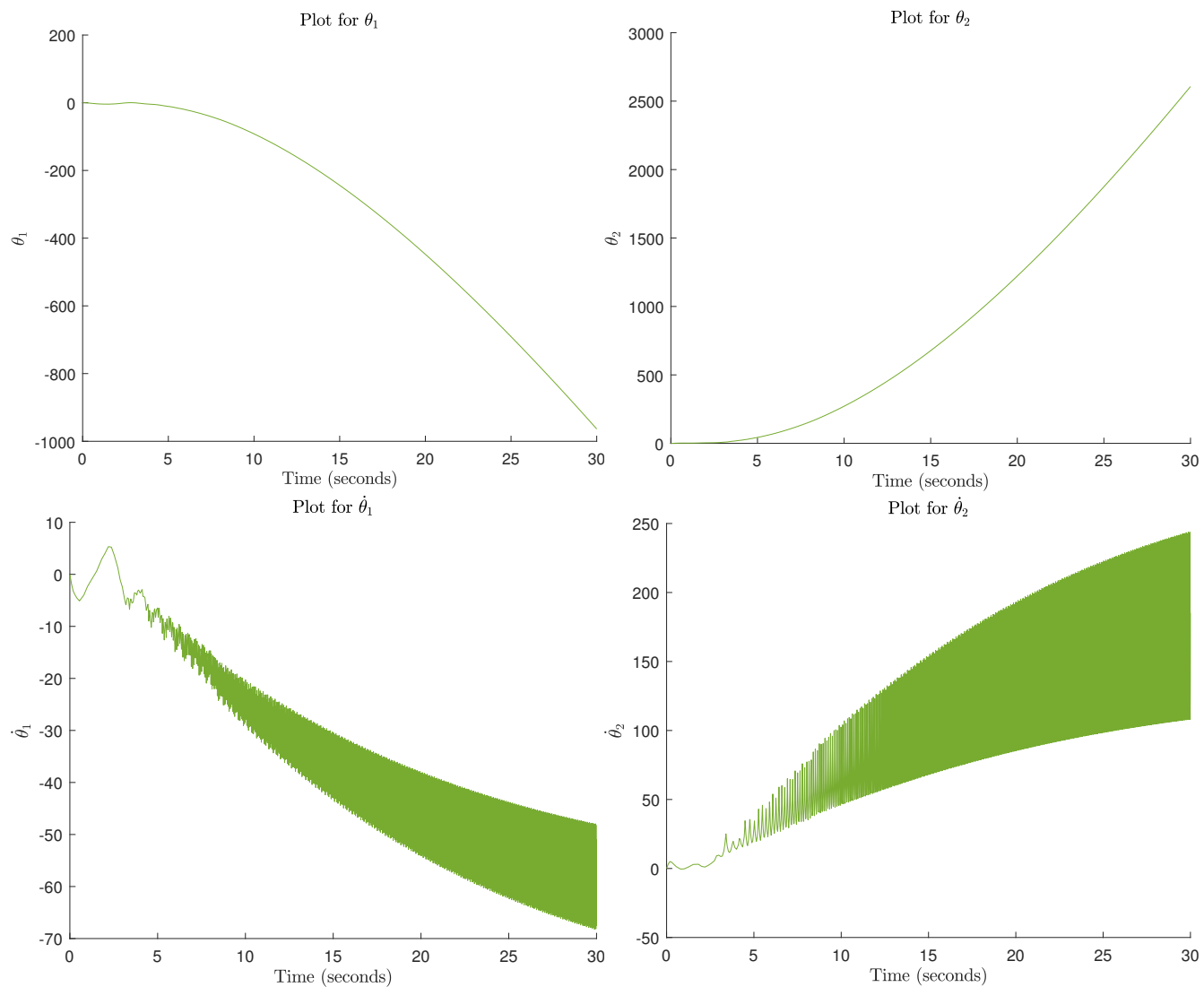


Figure 9: Joint angle and velocity plots for $x(0) = \begin{bmatrix} 0 & \frac{\pi}{2} & 0 & 0 \end{bmatrix}^T$, $\tau_1 = 0$, $\tau_2 = 5$

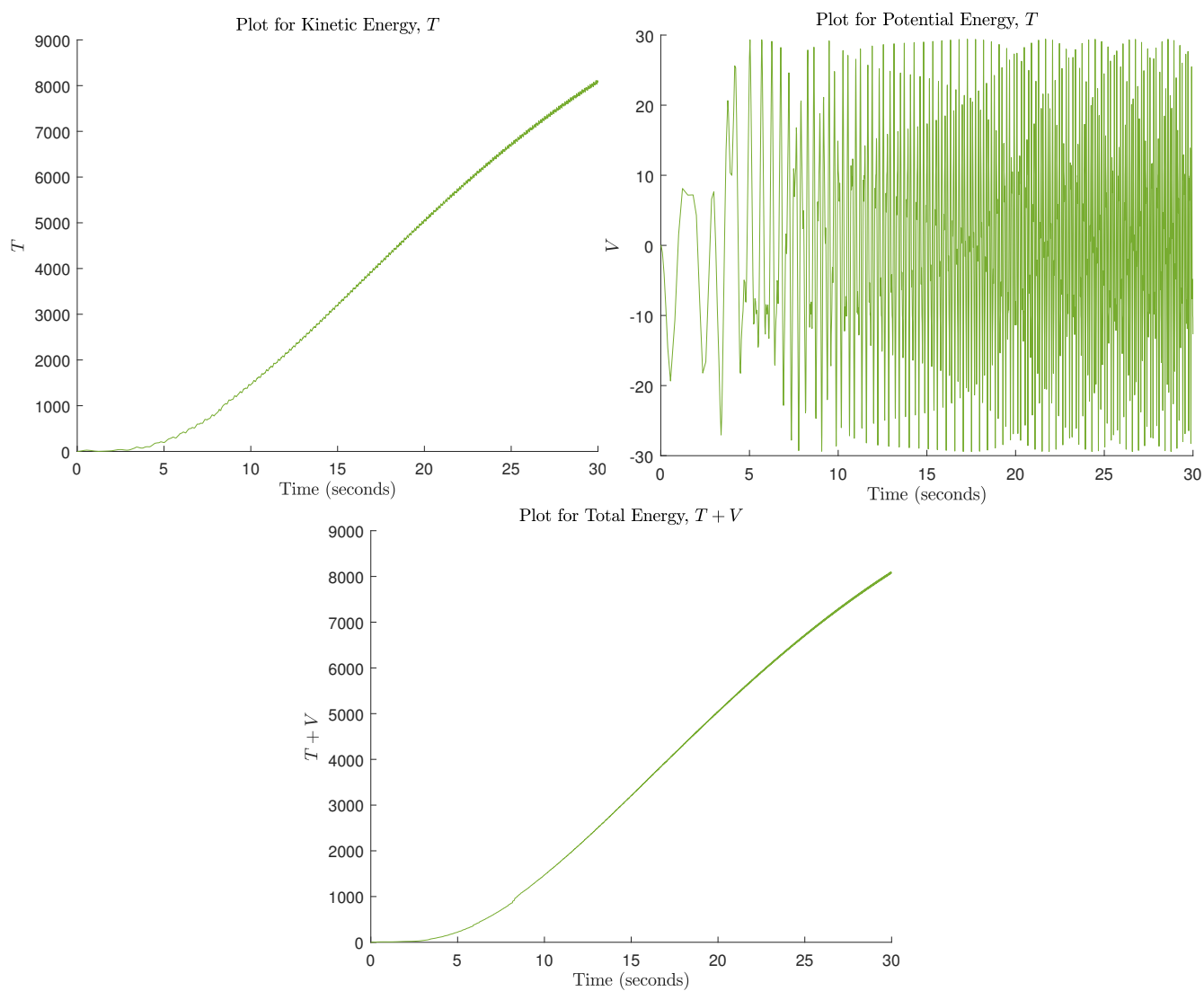


Figure 10: Energy plots for $x(0) = \begin{bmatrix} 0 & \frac{\pi}{2} & 0 & 0 \end{bmatrix}^T$, $\tau_1 = 0$, $\tau_2 = 5$

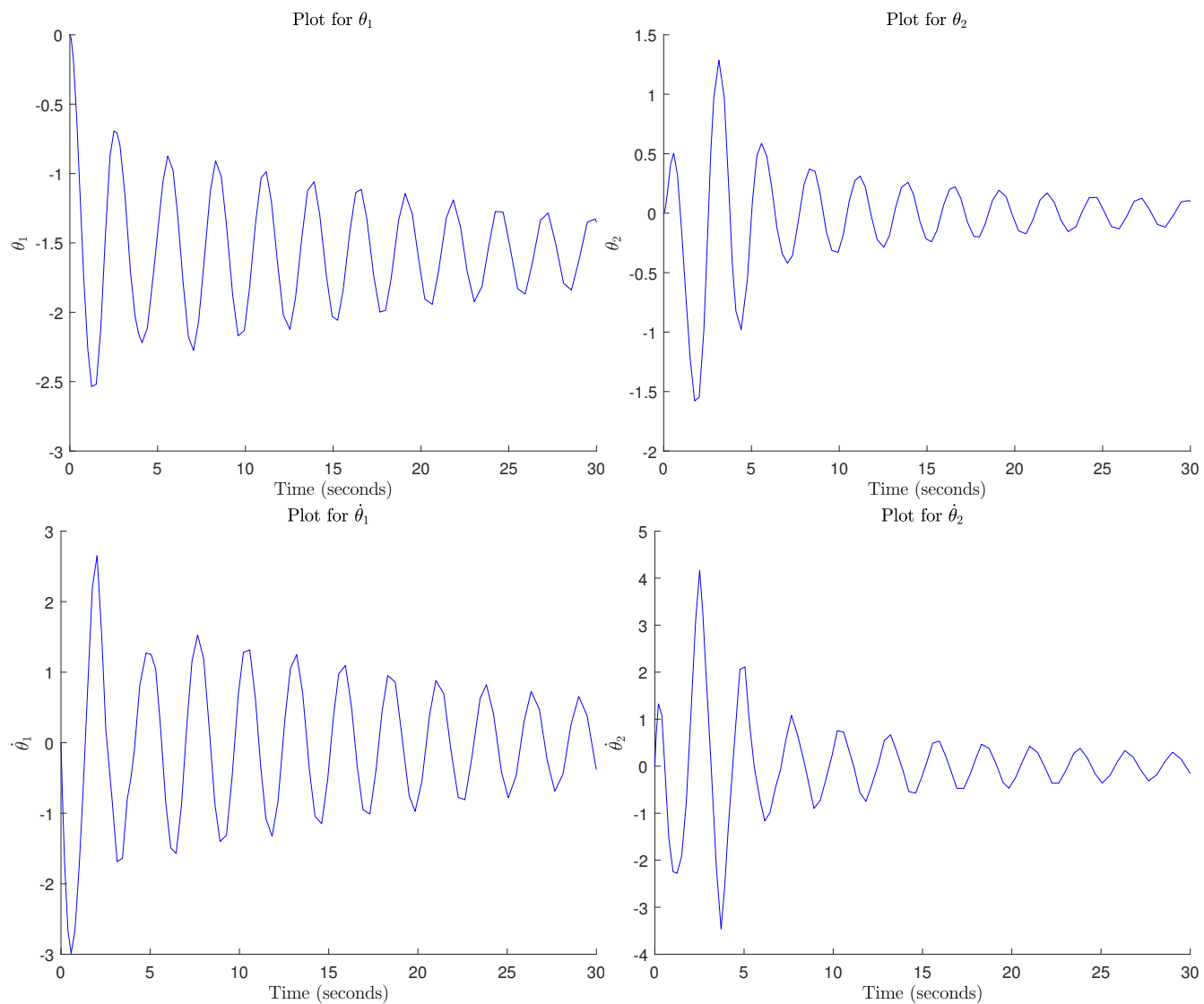


Figure 11: Joint angle and velocity plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = -0.5\dot{\theta}_1$, $\tau_2 = -0.5\dot{\theta}_2$

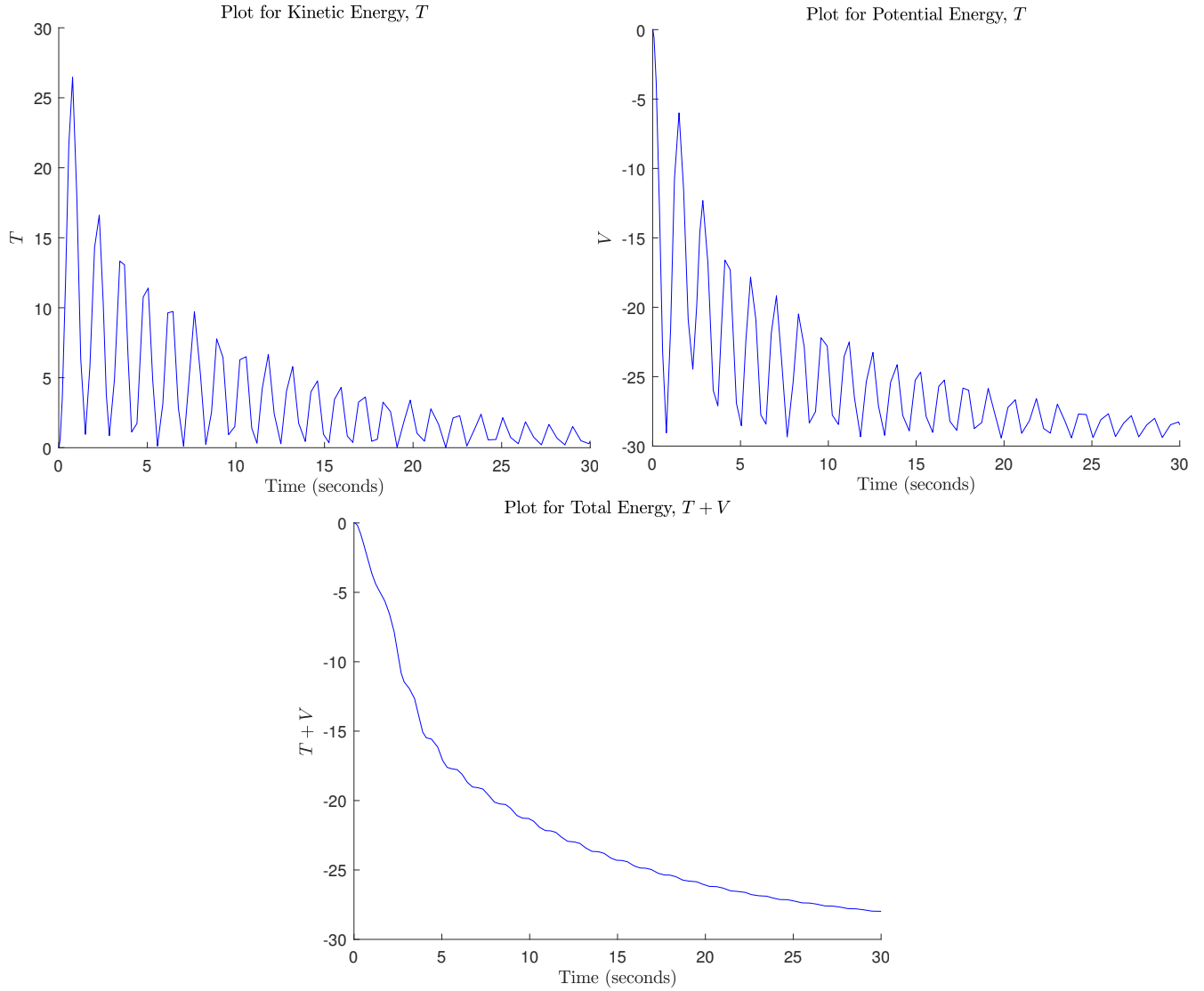


Figure 12: Energy plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = -0.5\dot{\theta}_1$, $\tau_2 = -0.5\dot{\theta}_2$

2 Closed-Loop Controller Implementation

2.1 Joint Space Control

The control law for this controller is

$$\underbrace{\mathbf{u}}_{\text{generalized motor torques}} = \underbrace{G(\mathbf{q})}_{\substack{\text{gravity} \\ \text{and other} \\ \text{potential energy terms}}} + \begin{bmatrix} K_p & K_v \end{bmatrix} \begin{bmatrix} \mathbf{q}_d - \mathbf{q} \\ -\dot{\mathbf{q}} \end{bmatrix}$$

$$\mathbf{u} = G(\mathbf{q}) + K_p(\mathbf{q}_d - \mathbf{q}) - K_v \underbrace{\dot{\mathbf{q}}}_{\substack{\text{constant} \\ \text{set-point}}}$$

where we set (for this question):

$$\begin{aligned}x(0) &= \begin{bmatrix} \frac{\pi}{2} \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ \mathbf{q}_d &= \begin{bmatrix} 0 \\ \frac{\pi}{2} \end{bmatrix} \\ K_p &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ K_v &= \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\end{aligned}$$

The simulation results for this run can be found in section 2.1.3.

2.1.1 Simulink block diagrams

The robot block in Figure 1 was made into a subsystem and connected to a PD + Gravity controller block, shown in Figure 13 below:

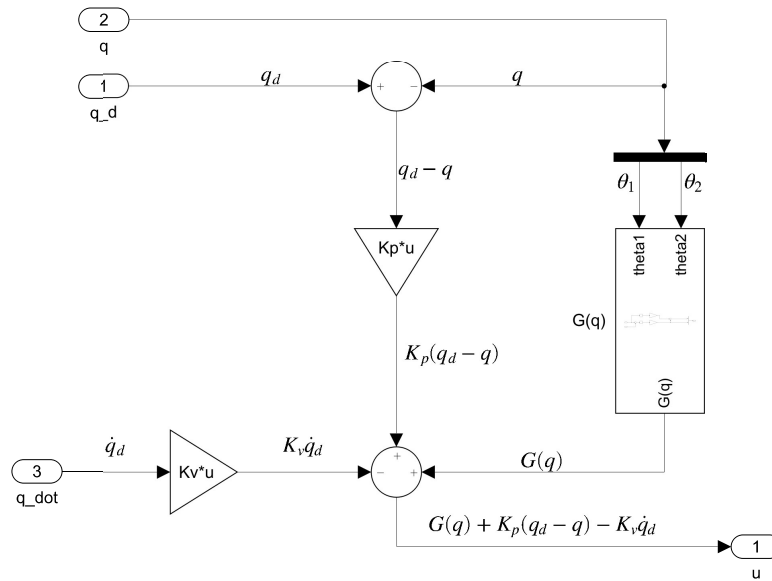


Figure 13: Simulink Block Diagram for the PD+Gravity Controller.

The overall system is shown in Figure 14:

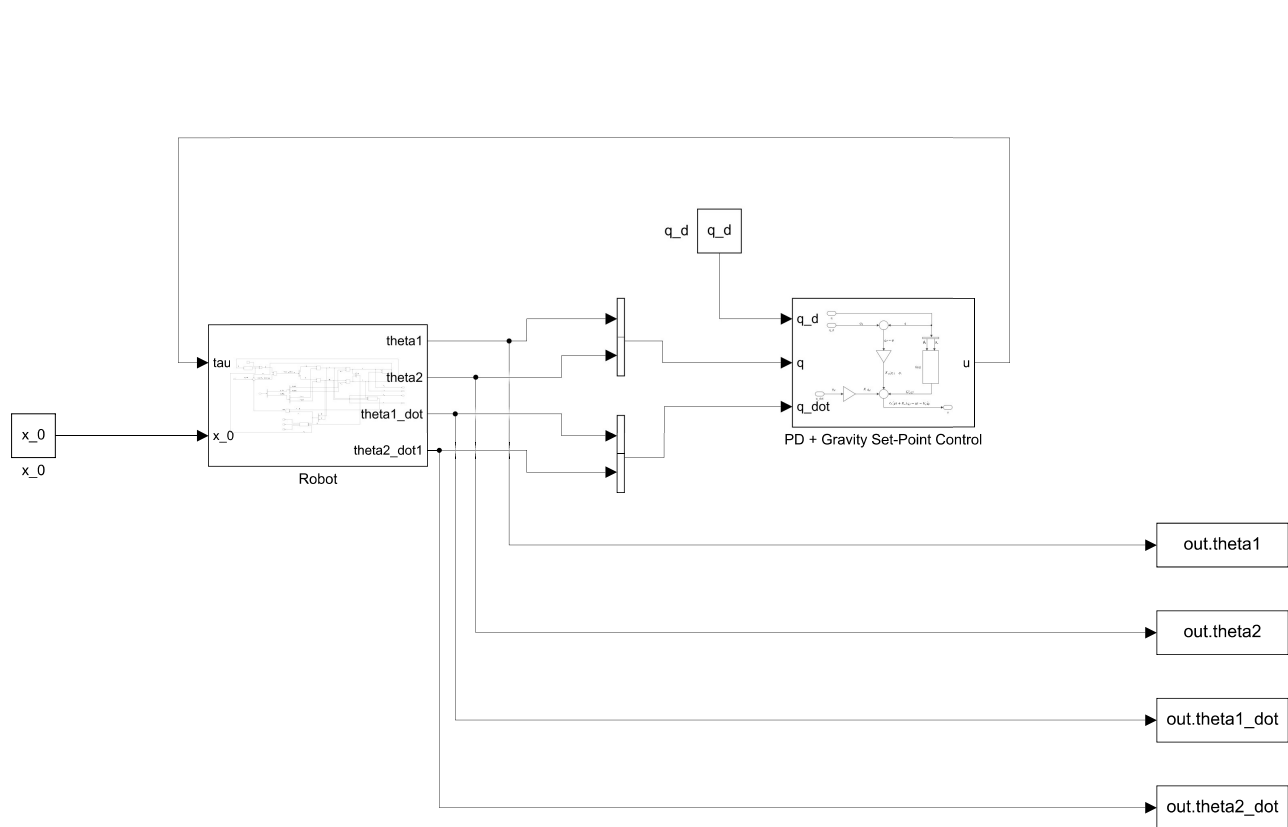


Figure 14: Simulink Block Diagram for the robot connected to the PD+Gravity controller.

2.1.2 MATLAB code

As in Listing 1, and Listing 2, a MATLAB function was written to automate simulation and saves the plots as .fig (and .pdf) files. The code is shown in Listing 3:

```

1 function plot_asn5q2a()
2     % Run simulation
3     simOut = sim('asn5q2');
4
5     % Take output values from Simulink
6     theta1=simOut.get('theta1');
7     theta2=simOut.get('theta2');
8

```

```

9      % Plots theta1
10     figure;
11     hold on;
12     view(2);
13     title('Plot for $\theta_1$', 'Interpreter', 'latex');
14     xlabel('Time (seconds)', 'Interpreter', 'latex');
15     ylabel('$\theta_1$', 'Interpreter', 'latex');
16     plot(theta1, 'Color', '#7E2F8E');
17     saveas(gcf, 'q2a.t1.fig'); % saves figure as .fig
18     saveas(gcf, 'q2a.t1', 'eps'); % saves figure as .eps (for preparing text)
19
20     % Plots theta2
21     figure;
22     hold on;
23     view(2);
24     title('Plot for $\theta_2$', 'Interpreter', 'latex');
25     xlabel('Time (seconds)', 'Interpreter', 'latex');
26     ylabel('$\theta_2$', 'Interpreter', 'latex');
27     plot(theta2, 'Color', '#7E2F8E');
28     saveas(gcf, 'q2a.t2.fig'); % saves figure as .fig
29     saveas(gcf, 'q2a.t2', 'eps'); % saves figure as .eps (for preparing text)
30 end

```

Listing 3: MATLAB code used to simulate the system for question 2(a).

2.1.3 Simulation results

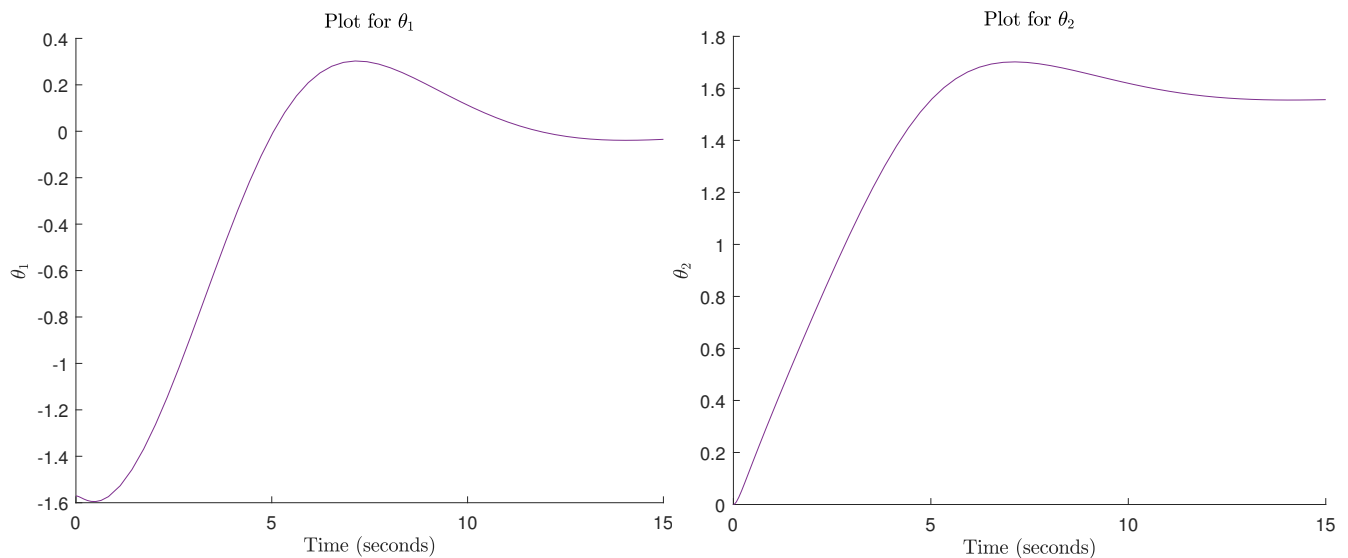


Figure 15: Joint angle plots for $x(0) = \begin{bmatrix} -\frac{\pi}{2} & 0 & 0 & 0 \end{bmatrix}^T$, $K_p = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $K_v = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, $q_d = \begin{bmatrix} 0 & \frac{\pi}{2} \end{bmatrix}^T$

2.2 Task Space Control

The control law for the Task Space Stiffness Controller is:

$$\mathbf{u} = G(\mathbf{q}) + \underbrace{\underline{J}^T}_{\substack{\text{transpose of} \\ \text{manipulator} \\ \text{Jacobian}}}(\mathbf{q}) \begin{bmatrix} \underline{\mathbf{f}}_n \\ \underline{\boldsymbol{\tau}}_n \end{bmatrix},$$

where

$$\begin{bmatrix} \underline{\mathbf{f}}_n \\ \underline{\boldsymbol{\tau}}_n \end{bmatrix} = K_P \begin{bmatrix} \underline{\mathbf{q}}_d - \underline{\mathbf{q}}_n \\ \underline{\mathbf{e}} \end{bmatrix} + K_V \begin{bmatrix} \dot{\underline{\mathbf{q}}}_d - \dot{\underline{\mathbf{q}}}_n \\ \underline{\boldsymbol{\omega}}_d - \underline{\boldsymbol{\omega}}_n \end{bmatrix}, \quad (2)$$

and

$$\underline{\mathbf{q}}_n = \underline{\mathbf{q}}_2(\theta) = \begin{bmatrix} l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \end{bmatrix}$$

For a two-link planar RR manipulator, we have that:

$$\begin{bmatrix} \underline{\mathbf{q}}_2 \\ \theta \end{bmatrix} = \underbrace{\begin{bmatrix} l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\ l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \\ \theta_1 + \theta_2 \end{bmatrix}}_{\mathbf{f}(\mathbf{q})}$$

Differentiating this expression w.r.t time gives:

$$\begin{aligned} \begin{bmatrix} \dot{\underline{\mathbf{q}}}_2 \\ \dot{\omega} \end{bmatrix} &= \frac{\partial \mathbf{f}}{\partial \mathbf{q}} \dot{\mathbf{q}} \\ &= \underbrace{\begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \\ 1 & 1 \end{bmatrix}}_{\text{Jacobian } \underline{J}} \dot{\mathbf{q}}, \end{aligned}$$

from which we can get:

$$\begin{aligned} \dot{\underline{\mathbf{q}}}_n = \dot{\underline{\mathbf{q}}}_2 &= \begin{bmatrix} [-l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2)]\dot{\theta}_1 + [-l_2 \sin(\theta_1 + \theta_2)]\dot{\theta}_2 \\ [l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)]\dot{\theta}_1 + [l_2 \cos(\theta_1 + \theta_2)]\dot{\theta}_2 \end{bmatrix} \\ \underline{\boldsymbol{\omega}}_n = \underline{\boldsymbol{\omega}}_2 &= \begin{bmatrix} 1 & 1 \end{bmatrix} \end{aligned}$$

However, as this is a 2 DOF manipulator, we can rewrite Equation 2 as:

$$\begin{bmatrix} \underline{\mathbf{f}}_n \\ \underline{\boldsymbol{\tau}}_n \end{bmatrix} = K_P \begin{bmatrix} \underline{\mathbf{q}}_d - \underline{\mathbf{q}}_n \end{bmatrix} + K_V \begin{bmatrix} \dot{\underline{\mathbf{q}}}_d - \dot{\underline{\mathbf{q}}}_n \end{bmatrix},$$

which makes the control law:

$$\begin{aligned}
 \mathbf{u} &= G(\mathbf{q}) + \underline{J}^T(\mathbf{q}) \begin{bmatrix} \underline{\mathbf{f}}_n \\ \underline{\boldsymbol{\tau}}_n \end{bmatrix} \\
 &= G(\mathbf{q}) + \underline{J}^T(\mathbf{q}) (K_P [\underline{\mathbf{q}}_d - \underline{\mathbf{q}}_2] + K_V [\dot{\underline{\mathbf{q}}}_d - \dot{\underline{\mathbf{q}}}_2]) \\
 &= G(\mathbf{q}) + \underline{J}^T(\mathbf{q}) (K_P \begin{bmatrix} o_{d1} - l_1 \cos(\theta_1) - l_2 \cos(\theta_1 + \theta_2) \\ o_{d2} - l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) \end{bmatrix} \\
 &\quad + K_V \begin{bmatrix} \dot{o}_{d1} - ([-l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2)]\dot{\theta}_1 + [-l_2 \sin(\theta_1 + \theta_2)]\dot{\theta}_2) \\ \dot{o}_{d2} - ([l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2)]\dot{\theta}_1 + [l_2 \cos(\theta_1 + \theta_2)]\dot{\theta}_2) \end{bmatrix}),
 \end{aligned}$$

and the Jacobian:

$$\underline{J} = \begin{bmatrix} -l_1 \sin(\theta_1) - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix}$$