ELEC 442 101

Introduction to Robotics

Assignment 5

Zhi Chuen Tan **Hubert Shim**

65408361 **??????**

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 generalizaed manipulator dynamics equation of the form:

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

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

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

where

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

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

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

$$C(\boldsymbol{q},\dot{\boldsymbol{q}}) = \begin{bmatrix}-2m_2l_1l_2(\sin\theta_2)\dot{\theta_2} & -m_2l_1l_2(\sin\theta_2)\dot{\theta_2}\\ m_2l_1l_2(\sin\theta_2)\dot{\theta_1} & 0\end{bmatrix}$$

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

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

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

$$\Rightarrow E_T = T + V$$

1.1 Simulation

1.1.1 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 LaTeXtypesetting, which makes these diagrams vector diagrams, i.e. they can be zoomed-in without significant aliasing):

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

    % Run simulation
    simOut = sim('asn5q1');

    % Take output values from Simulink
    theta1=simOut.get('theta1');
    theta2=simOut.get('theta2');
    theta1_dot=simOut.get('theta1_dot');
    theta2_dot=simOut.get('theta2_dot');
    T=simOut.get('T');
    V=simOut.get('V');
    E_total=simOut.get('E_total');

    % Plots theta1
    figure;
    hold on;
    view(2);
    title('Plot for $\theta_{1}$', 'Interpreter', 'latex');
    xlabel('Time (seconds)', 'Interpreter', 'latex');
    ylabel('$\theta_{1}$', 'Interpreter', 'latex');
    plot(theta1, 'Color', '#A2142F');
    saveas(gcf, 'q1a_t1.fig'); % saves figure as .fig
    saveas(gcf, 'q1a_t1', 'epsc'); % saves figure as .eps (for preparing text)

    % Plots theta2
    figure;
    hold on;
    view(2);
    title('Plot for $\theta_{2}$', 'Interpreter', 'latex');
    xlabel('Time (seconds)', 'Interpreter', 'latex');
```

```
    ylabel('$\theta_{2}$', 'Interpreter', 'latex');
    plot(theta2, 'Color', '#A2142F');
    saveas(gcf, 'q1a_t2.fig'); % saves figure as .fig
    saveas(gcf, 'q1a_t2', 'epsc'); % saves figure as .eps (for preparing text)

    % Plots theta1_dot
    figure;
    hold on;
    view(2);
    title('Plot for $\dot{\theta}_{1}$', 'Interpreter', 'latex');
    xlabel('Time (seconds)', 'Interpreter', 'latex');
    ylabel('$\dot{\theta}_{1}$', 'Interpreter', 'latex');
    plot(theta1_dot, 'Color', '#A2142F');
    saveas(gcf, 'q1a_t1d.fig'); % saves figure as .fig
    saveas(gcf, 'q1a_t1d', 'epsc'); % saves figure as .eps (for preparing text)

    % Plots theta2_dot
    figure;
    hold on;
    view(2);
    title('Plot for $\dot{\theta}_{2}$', 'Interpreter', 'latex');
    xlabel('Time (seconds)', 'Interpreter', 'latex');
    ylabel('$\dot{\theta}_{2}$', 'Interpreter', 'latex');
    plot(theta2_dot, 'Color', '#A2142F');
    saveas(gcf, 'q1a_t2d.fig'); % saves figure as .fig
    saveas(gcf, 'q1a_t2d', 'epsc'); % saves figure as .eps (for preparing text)

    % Plots kinetic energy
    figure;
    hold on;
    view(2);
    title('Plot for Kinetic Energy, $T$', 'Interpreter', 'latex');
    xlabel('Time (seconds)', 'Interpreter', 'latex');
    ylabel('$T$', 'Interpreter', 'latex');
    plot(T, 'Color', '#A2142F');
    saveas(gcf, 'q1a_T.fig'); % saves figure as .fig
    saveas(gcf, 'q1a_T', 'epsc'); % saves figure as .eps (for preparing text)

    % Plots potential energy
    figure;
    hold on;
    view(2);
    title('Plot for Potential Energy, $T$', 'Interpreter', 'latex');
    xlabel('Time (seconds)', 'Interpreter', 'latex');
    ylabel('$V$', 'Interpreter', 'latex');
```

```
    plot(V, 'Color', '#A2142F');
    saveas(gcf, 'q1a_V.fig'); % saves figure as .fig
    saveas(gcf, 'q1a_V', 'epsc'); % saves figure as .eps (for preparing text)

    % Plots total energy
    figure;
    hold on;
    view(2);
    title('Plot for Total Energy, $T+V$', 'Interpreter', 'latex');
    xlabel('Time (seconds)', 'Interpreter', 'latex');
    ylabel('$T+V$', 'Interpreter', 'latex');
    plot(E_total, 'Color', '#A2142F');
    saveas(gcf, 'q1a_total.fig'); % saves figure as .fig
    saveas(gcf, 'q1a_total', 'epsc'); % saves figure as .eps (for preparing text)
end
```

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

```
function run_sim_asn5q1c(x_0_in)
    % Initialize constants
    x_0 = x_0_in;

    l1=1;l2=1;m1=1;m2=1;g=9.81;

    % Run simulation
    simOut = sim('asn5q1c');

    % Take output values from Simulink
    theta1=simOut.get('theta1');
    theta2=simOut.get('theta2');
    theta1_dot=simOut.get('theta1_dot');
    theta2_dot=simOut.get('theta2_dot');
    T=simOut.get('T');
    V=simOut.get('V');
    E_total=simOut.get('E_total');

    % Plots theta1
    figure;
    hold on;
    view(2);
    title('Plot for $\theta_{1}$', 'Interpreter', 'latex');
    xlabel('Time (seconds)', 'Interpreter', 'latex');
    ylabel('$\theta_{1}$', 'Interpreter', 'latex');
    plot(theta1, 'blue');
```

```
27      saveas(gcf, 'q1c_t1.fig'); % saves figure as .fig
28      saveas(gcf, 'q1c_t1', 'epsc'); % 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', 'epsc'); % 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, 'q1c_t1d.fig'); % saves figure as .fig
50      saveas(gcf, 'q1c_t1d', 'epsc'); % 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, 'q1c_t2d.fig'); % saves figure as .fig
61      saveas(gcf, 'q1c_t2d', 'epsc'); % 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, 'q1c_T.fig'); % saves figure as .fig
```

```
72      saveas(gcf, 'q1c_T', 'epsc'); % 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, $T$', 'Interpreter', 'latex');
79      xlabel('Time (seconds)', 'Interpreter', 'latex');
80      ylabel('$V$', 'Interpreter', 'latex');
81      plot(V, 'blue');
82      saveas(gcf, 'q1c_V.fig'); % saves figure as .fig
83      saveas(gcf, 'q1c_V', 'epsc'); % 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, 'blue');
93      saveas(gcf, 'q1c_total.fig'); % saves figure as .fig
94      saveas(gcf, 'q1c_total', 'epsc'); % 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)

1.1.2 Simulation results

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

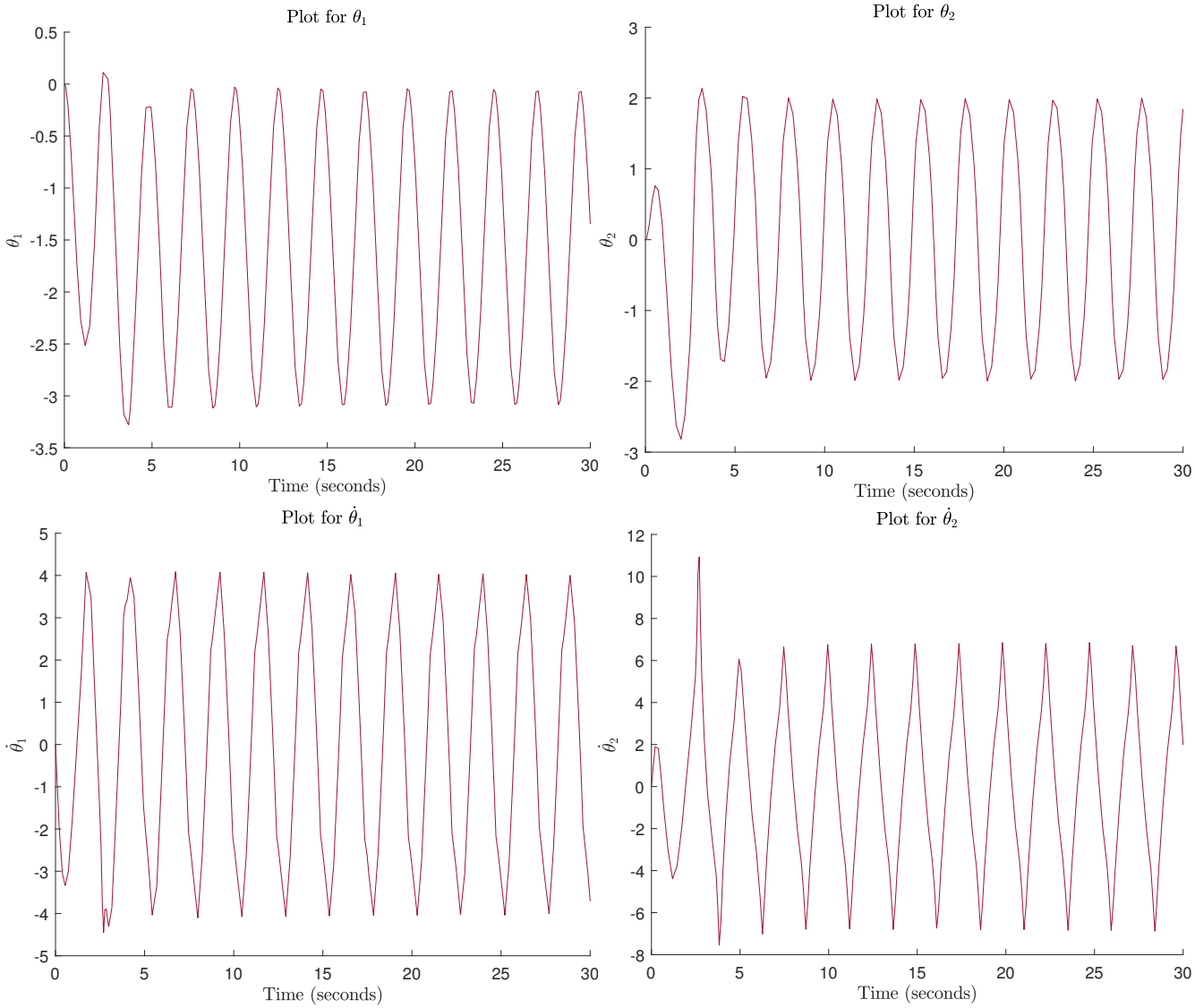

Figure 1: Joint angle and velocity plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = \tau_2 = 0$

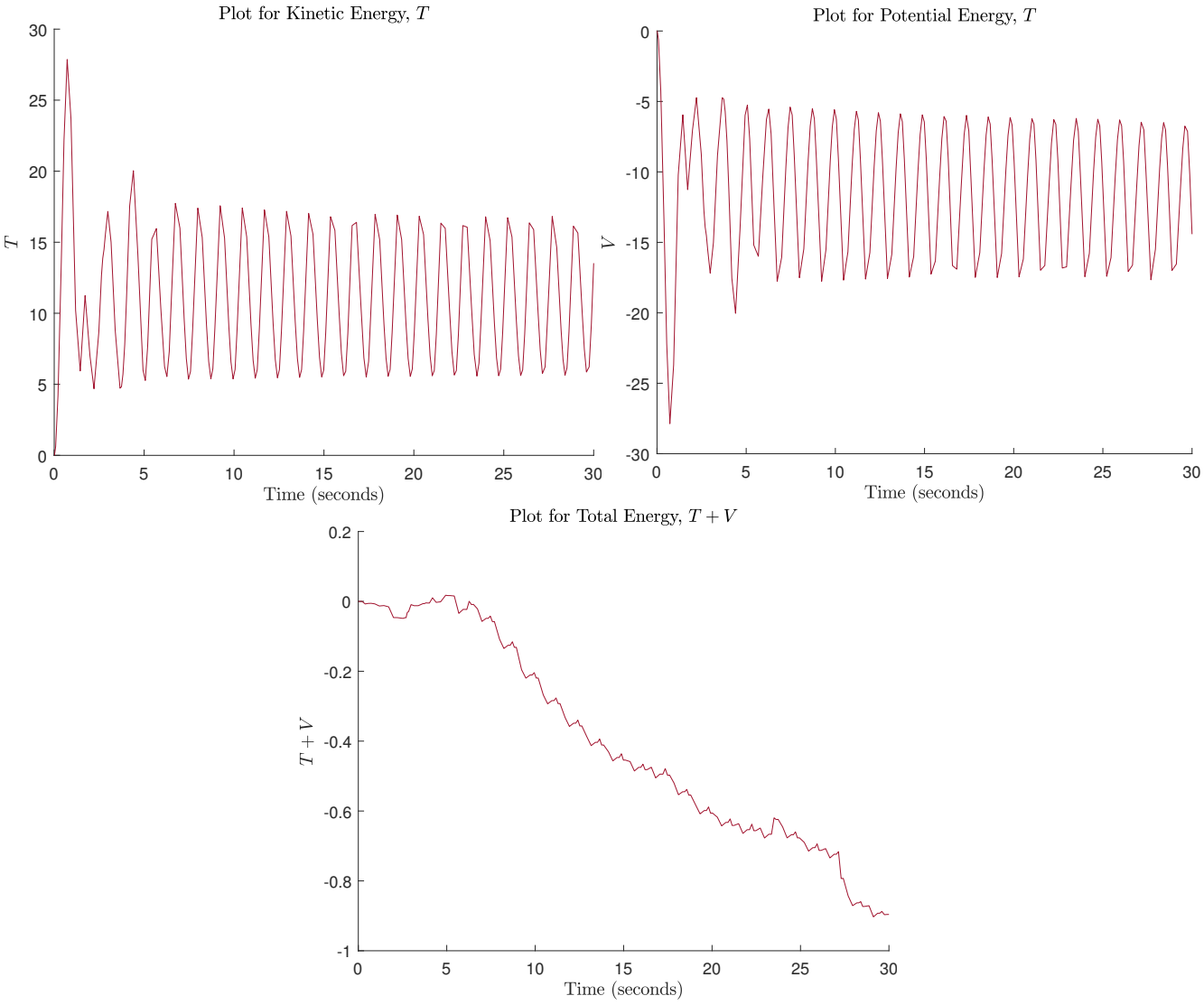

Figure 2: Energy plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = \tau_2 = 0$

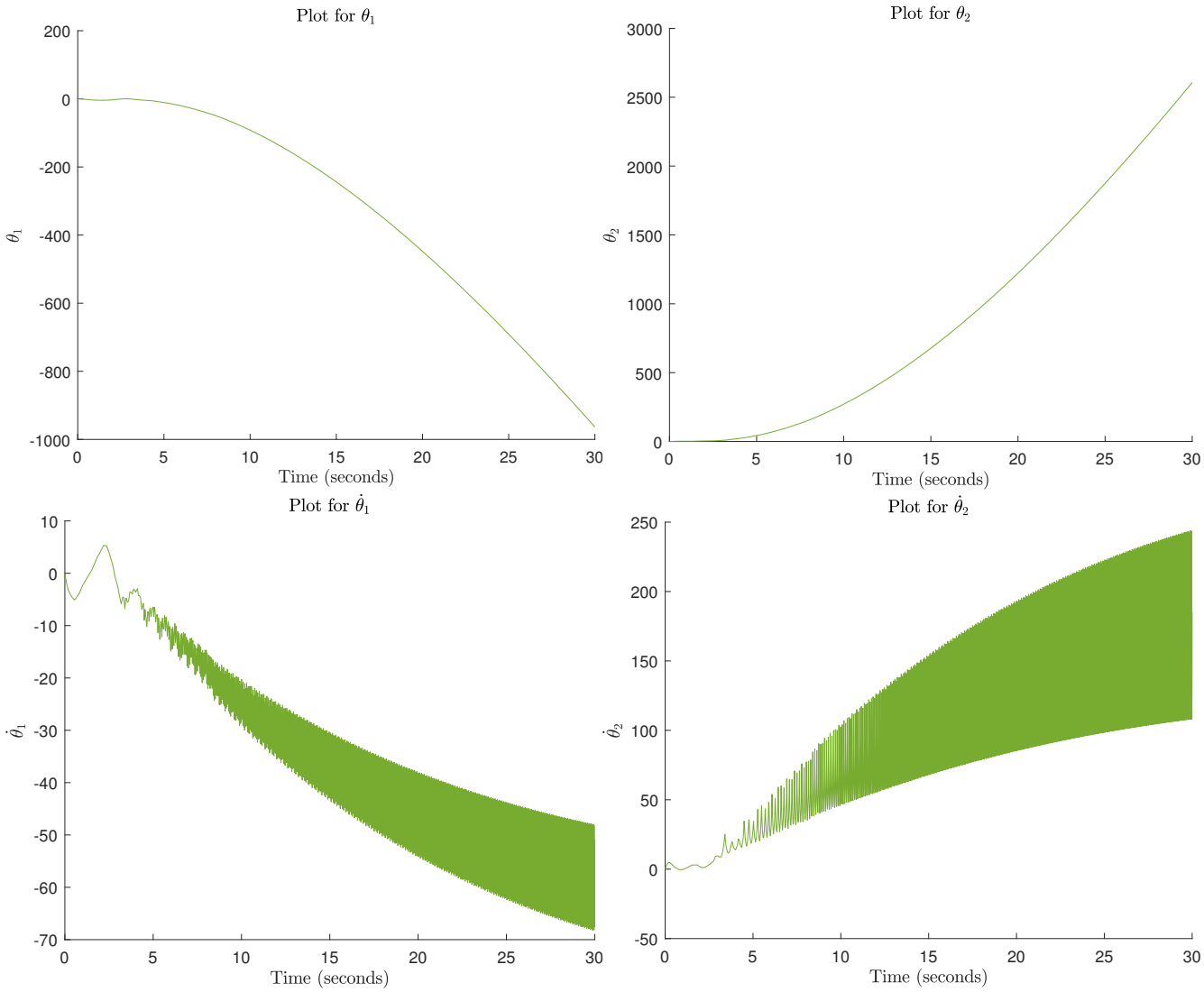

Figure 3: Joint angle and velocity plots for $x(0) = \begin{bmatrix} 0 & \frac{\pi}{2} & 0 & 0 \end{bmatrix}^T$, $\tau_1 = 0$, $\tau_2 = 5$

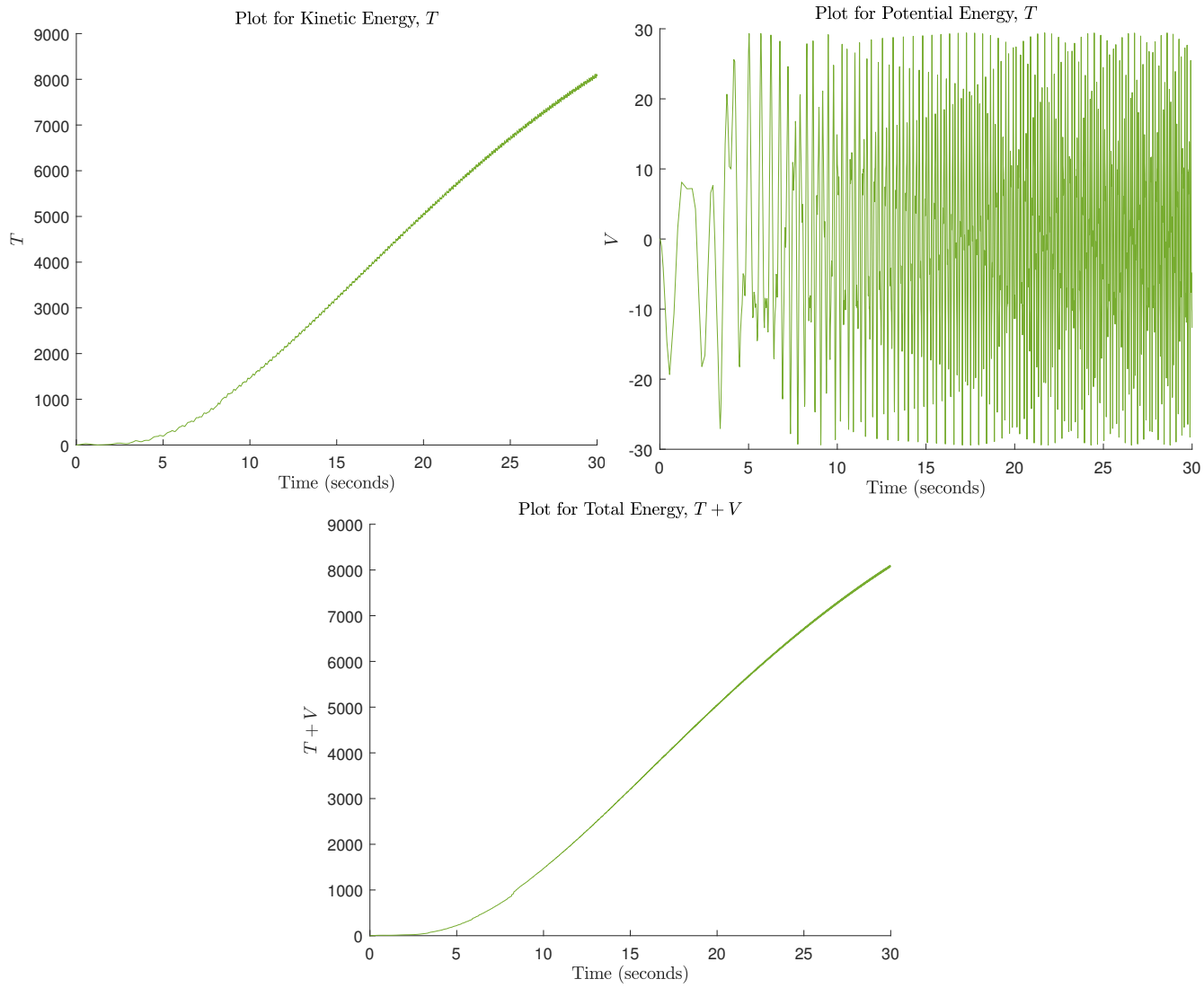

Figure 4: Energy plots for $x(0) = \begin{bmatrix} 0 & \frac{\pi}{2} & 0 & 0 \end{bmatrix}^T$, $\tau_1 = 0$, $\tau_2 = 5$

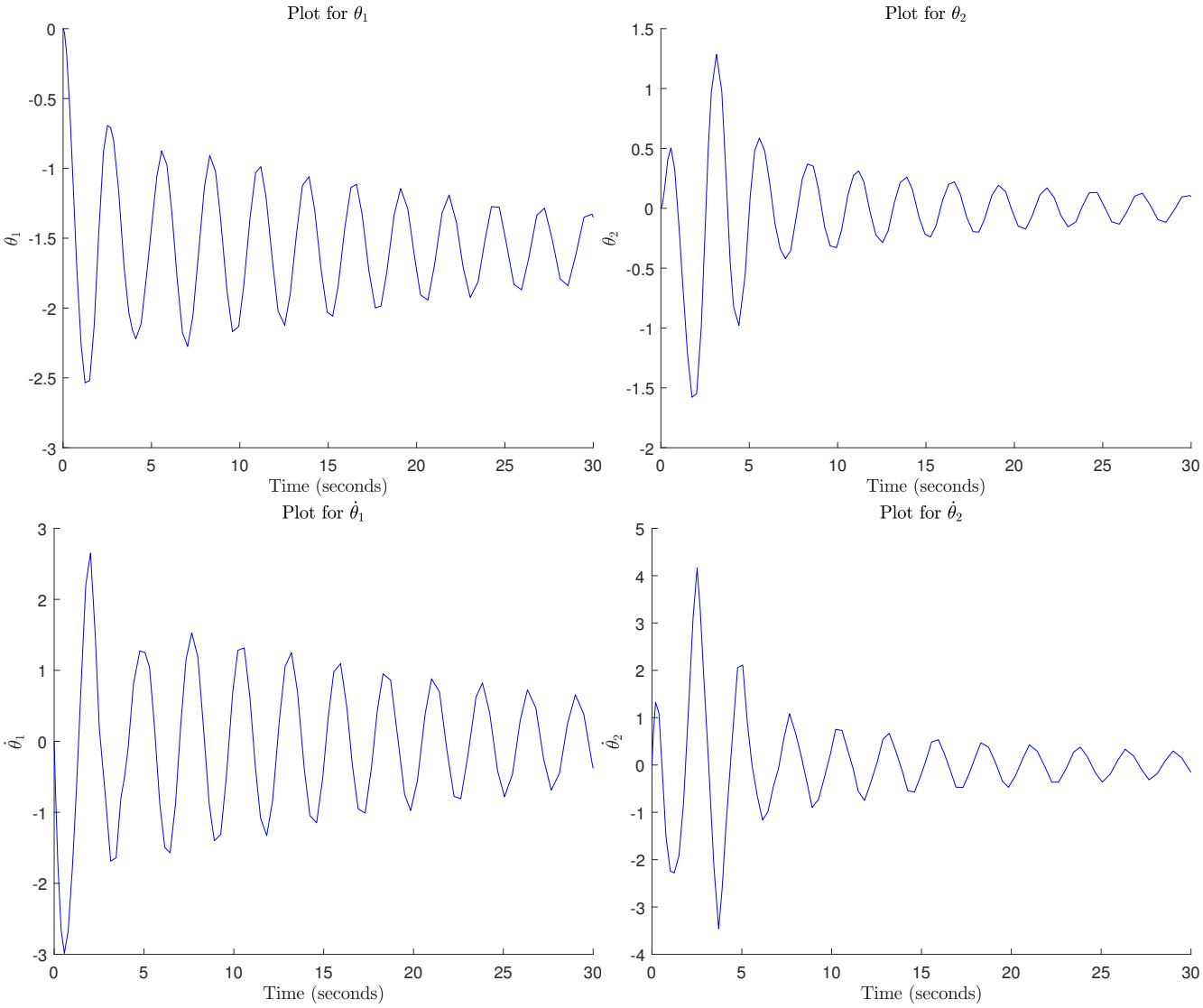

Figure 5: Joint angle and velocity plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = -0.5\dot{\theta}_1$, $\tau_2 = -0.5\dot{\theta}_2$

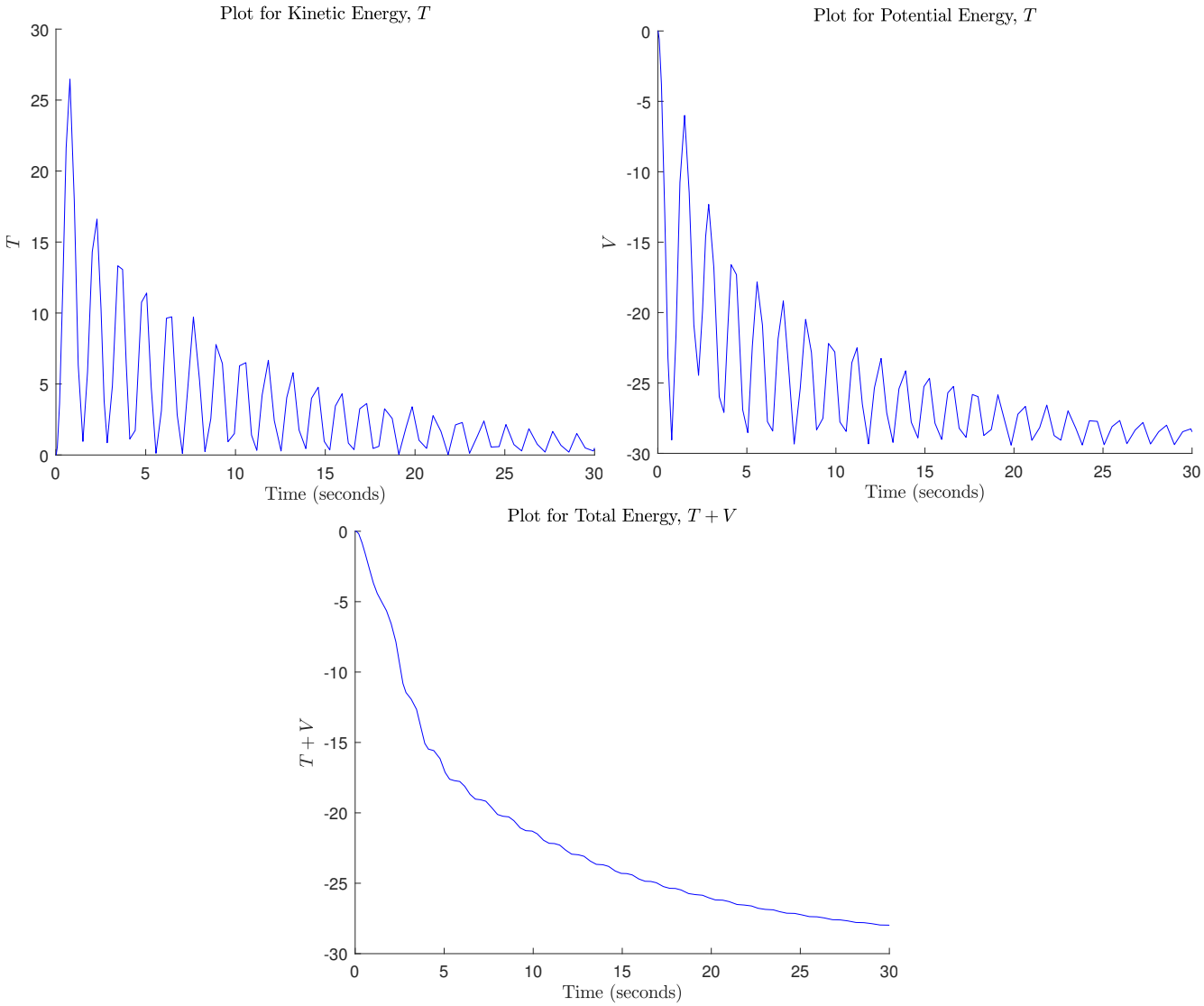

Figure 6: Energy plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = -0.5\dot{\theta}_1$, $\tau_2 = -0.5\dot{\theta}_2$

1.1.3 Simulink block diagrams

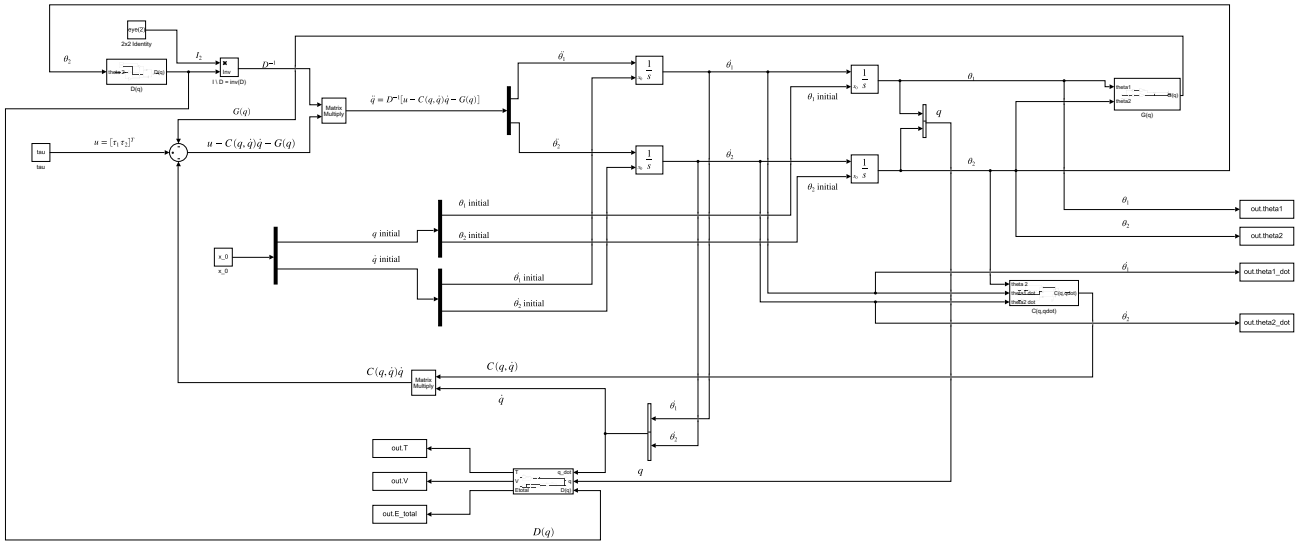

Figure 7: Simulink Block Diagram for the robot.

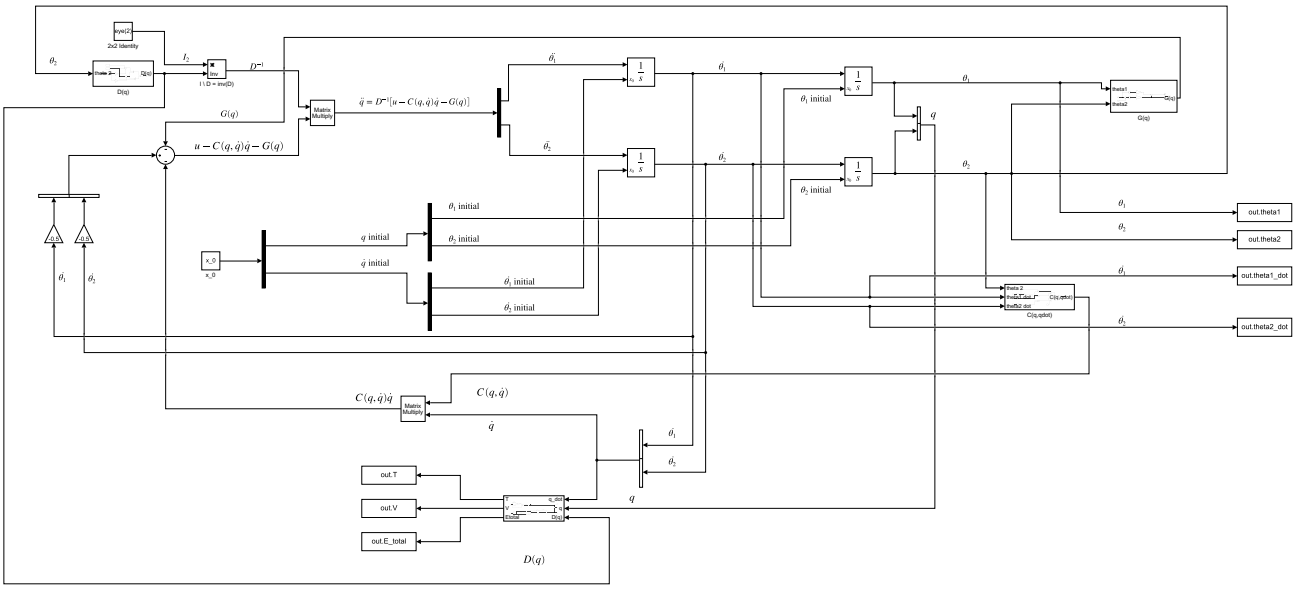

Figure 8: Simulink Block Diagram for the robot, used in part 1(c) to feed friction back into the system.)

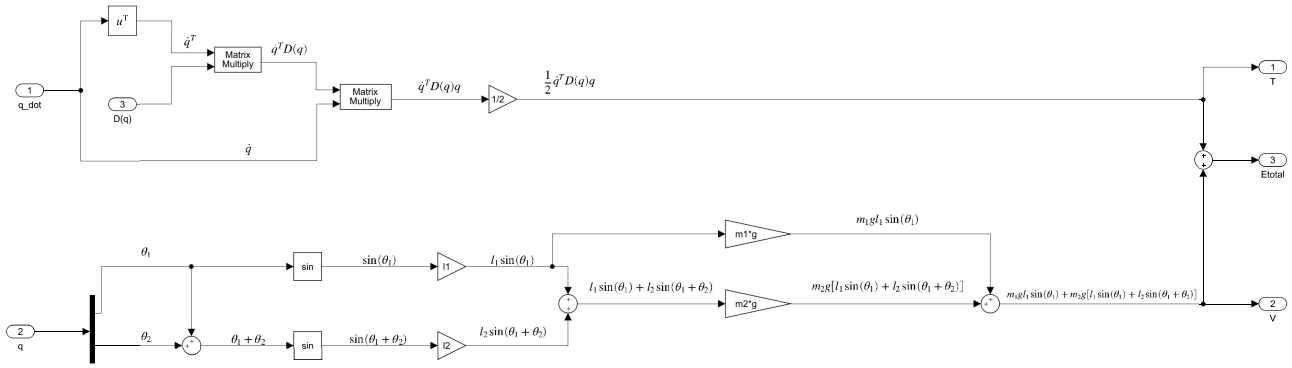

Figure 9: Simulink Block Diagram for the the subsystem used to calculate (potetial, kinetic, and total) energy.

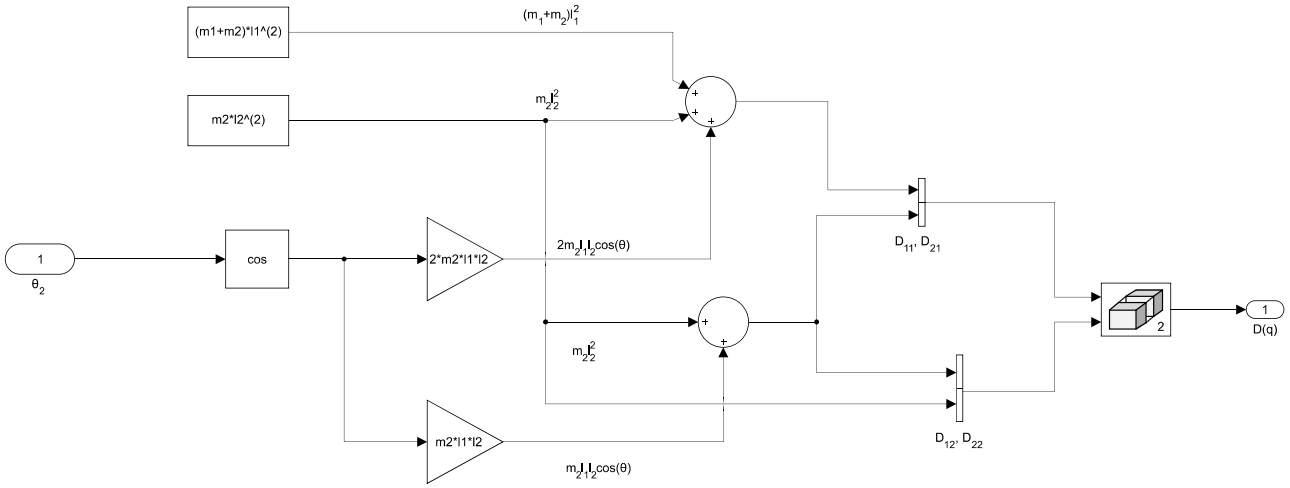

Figure 10: Simulink Block Diagram for the the subsystem used to calculate the $D(\boldsymbol{q})$ matrix.

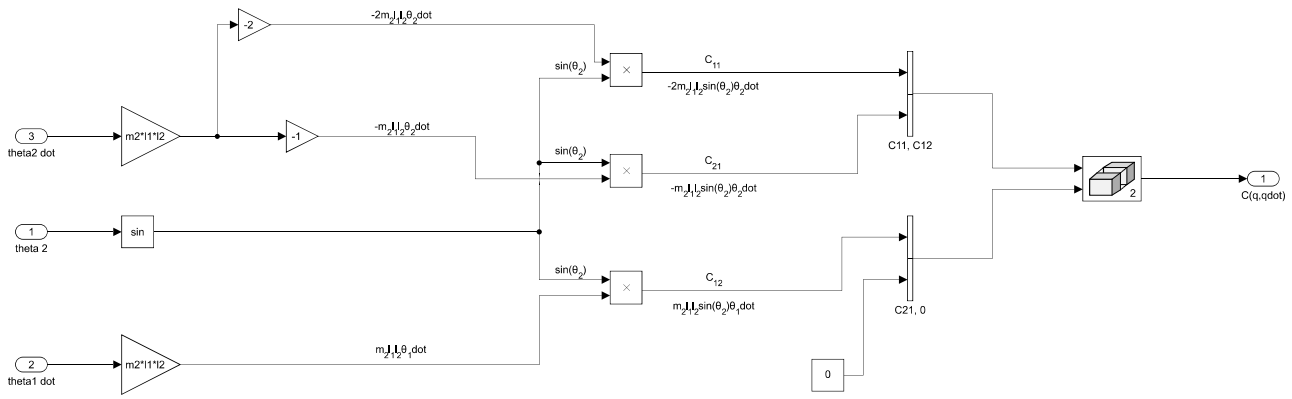

Figure 11: Simulink Block Diagram for the the subsystem used to calculate the $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ matrix.

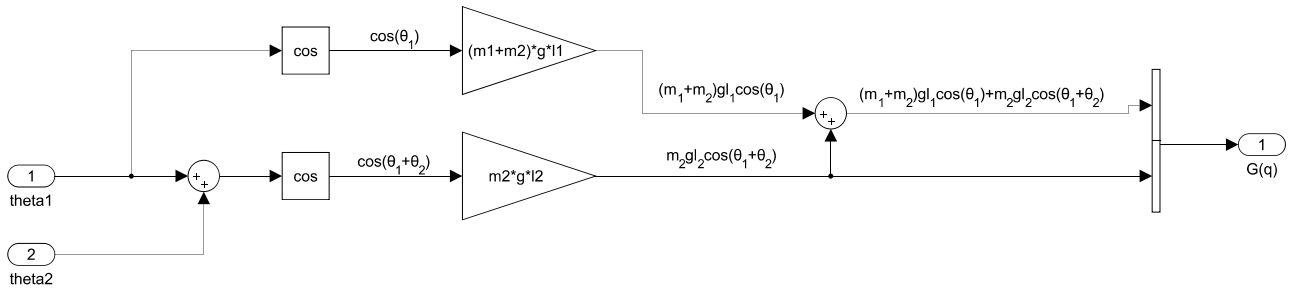

Figure 12: Simulink Block Diagram for the the subsystem used to calculate the $G(\boldsymbol{q})$ matrix.

2 Closed-Loop Controller Implementation

2.1 Joint Space Control

The control law for this controller is

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

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

where we set (for this question):

$$x(0) = \begin{bmatrix} \frac{\pi}{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{q}_d = \begin{bmatrix} 0 \\ \frac{\pi}{2} \end{bmatrix}$$

$$K_p = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$K_v = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

2.1.1 Simulink block diagrams

The robot block in Figure 7 was made into a subsystem and connected to a PD + Gravity controller block, shown in Figure 13 below:

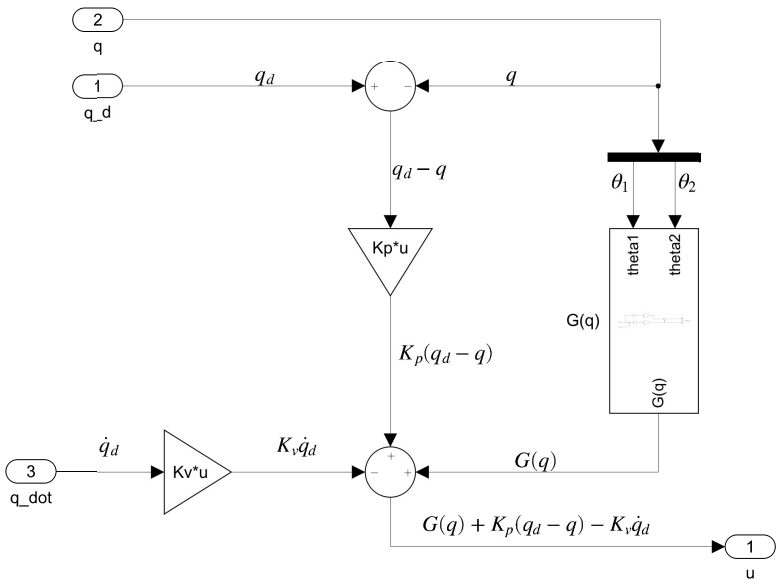

Figure 13: Simulink Block Diagram for the PD+Gravity Controller.

The overall system is shown in Figure 14:

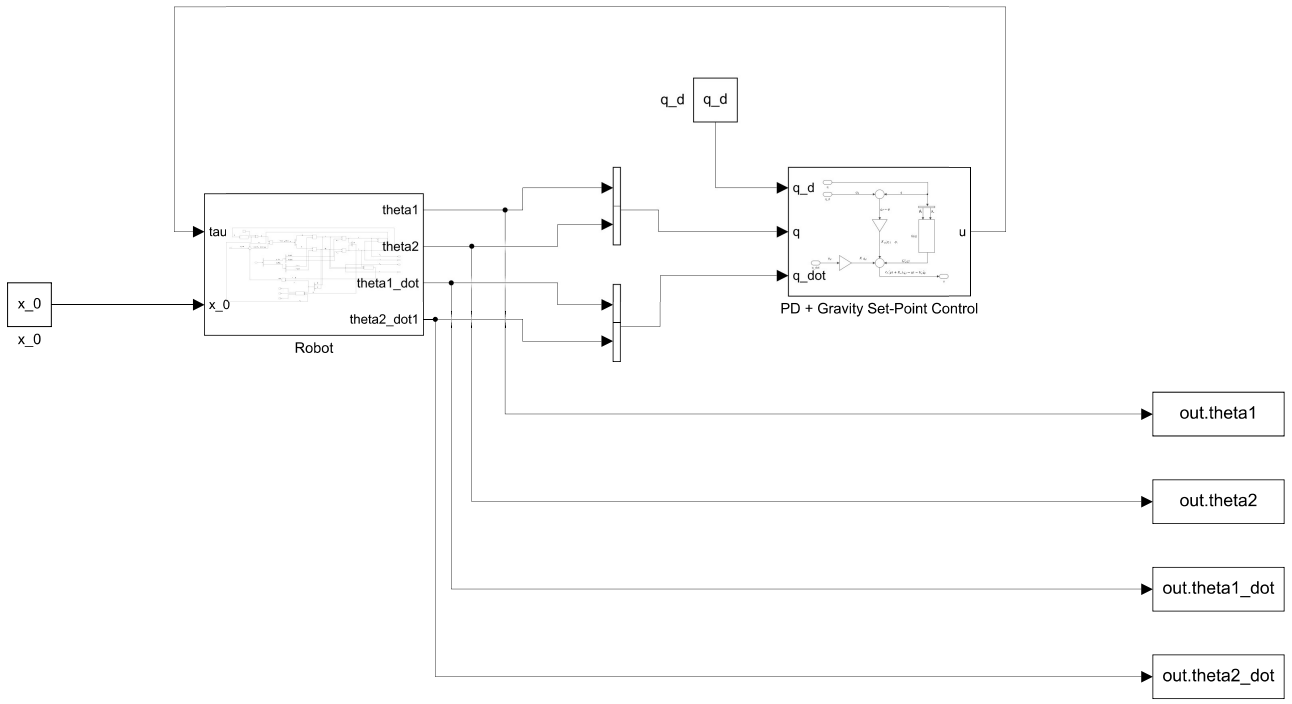

Figure 14: Simulink Block Diagram for the robot connected to the PD+Gravity controller.

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
```

```
    % Plots theta1
    figure;
    hold on;
    view(2);
    title('Plot for $\theta_{1}$', 'Interpreter', 'latex');
    xlabel('Time (seconds)', 'Interpreter', 'latex');
    ylabel('$\theta_{1}$', 'Interpreter', 'latex');
    plot(theta1, 'Color', '#7E2F8E');
    saveas(gcf, 'q2a_t1.fig'); % saves figure as .fig
    saveas(gcf, 'q2a_t1', 'epsc'); % saves figure as .eps (for preparing text)

    % Plots theta2
    figure;
    hold on;
    view(2);
    title('Plot for $\theta_{2}$', 'Interpreter', 'latex');
    xlabel('Time (seconds)', 'Interpreter', 'latex');
    ylabel('$\theta_{2}$', 'Interpreter', 'latex');
    plot(theta2, 'Color', '#7E2F8E');
    saveas(gcf, 'q2a_t2.fig'); % saves figure as .fig
    saveas(gcf, 'q2a_t2', 'epsc'); % saves figure as .eps (for preparing text)
end
```

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

2.1.3 Simulation results

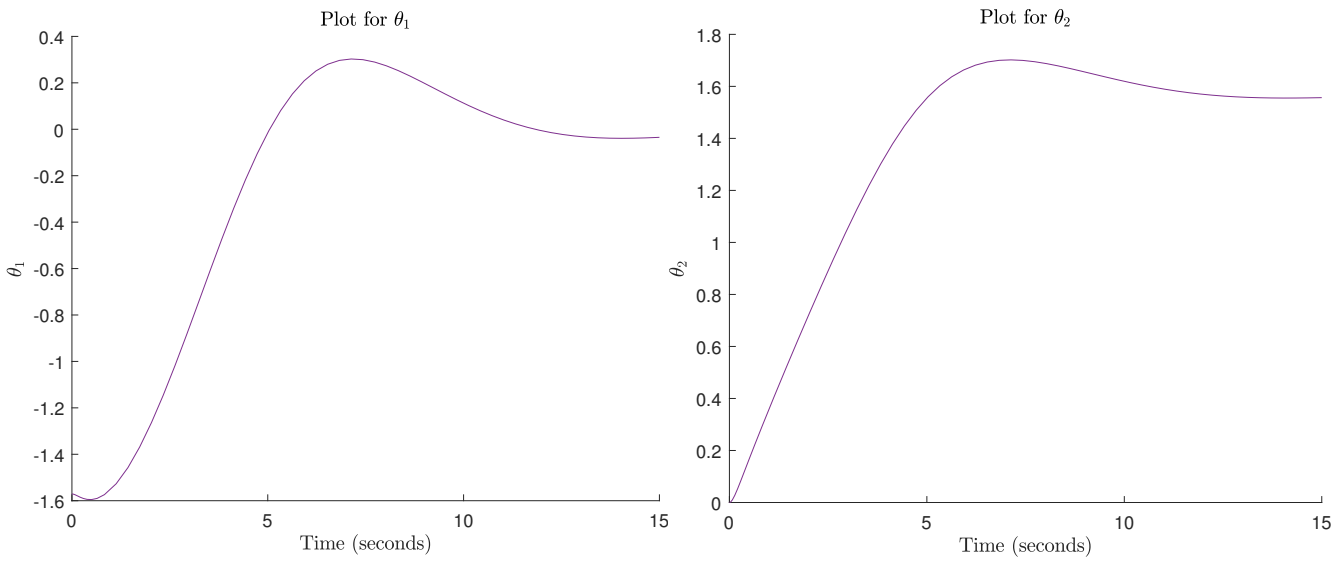

Figure 15: Joint angle plots for $x(0) = \begin{bmatrix} -\frac{\pi}{2} & 0 & 0 & 0 \end{bmatrix}^T$, $K_p = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $K_v = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$, $\boldsymbol{q}_d = \begin{bmatrix} 0 & \frac{\pi}{2} \end{bmatrix}^T$

2.2 Task Space Control