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:

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

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

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

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

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

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

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

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

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

$$\Rightarrow E_T = T + V$$

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');
```

```
    saveas(gcf, 'q1c_t1.fig'); % saves figure as .fig
    saveas(gcf, 'q1c_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, 'blue');
    saveas(gcf, 'q1c_t2.fig'); % saves figure as .fig
    saveas(gcf, 'q1c_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, 'blue');
    saveas(gcf, 'q1c_t1d.fig'); % saves figure as .fig
    saveas(gcf, 'q1c_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, 'blue');
    saveas(gcf, 'q1c_t2d.fig'); % saves figure as .fig
    saveas(gcf, 'q1c_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, 'blue');
    saveas(gcf, 'q1c_T.fig'); % saves figure as .fig
```

```
    saveas(gcf, 'q1c_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, 'blue');
    saveas(gcf, 'q1c_V.fig'); % saves figure as .fig
    saveas(gcf, 'q1c_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, 'blue');
    saveas(gcf, 'q1c_total.fig'); % saves figure as .fig
    saveas(gcf, 'q1c_total', 'epsc'); % saves figure as .eps (for preparing text)






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.

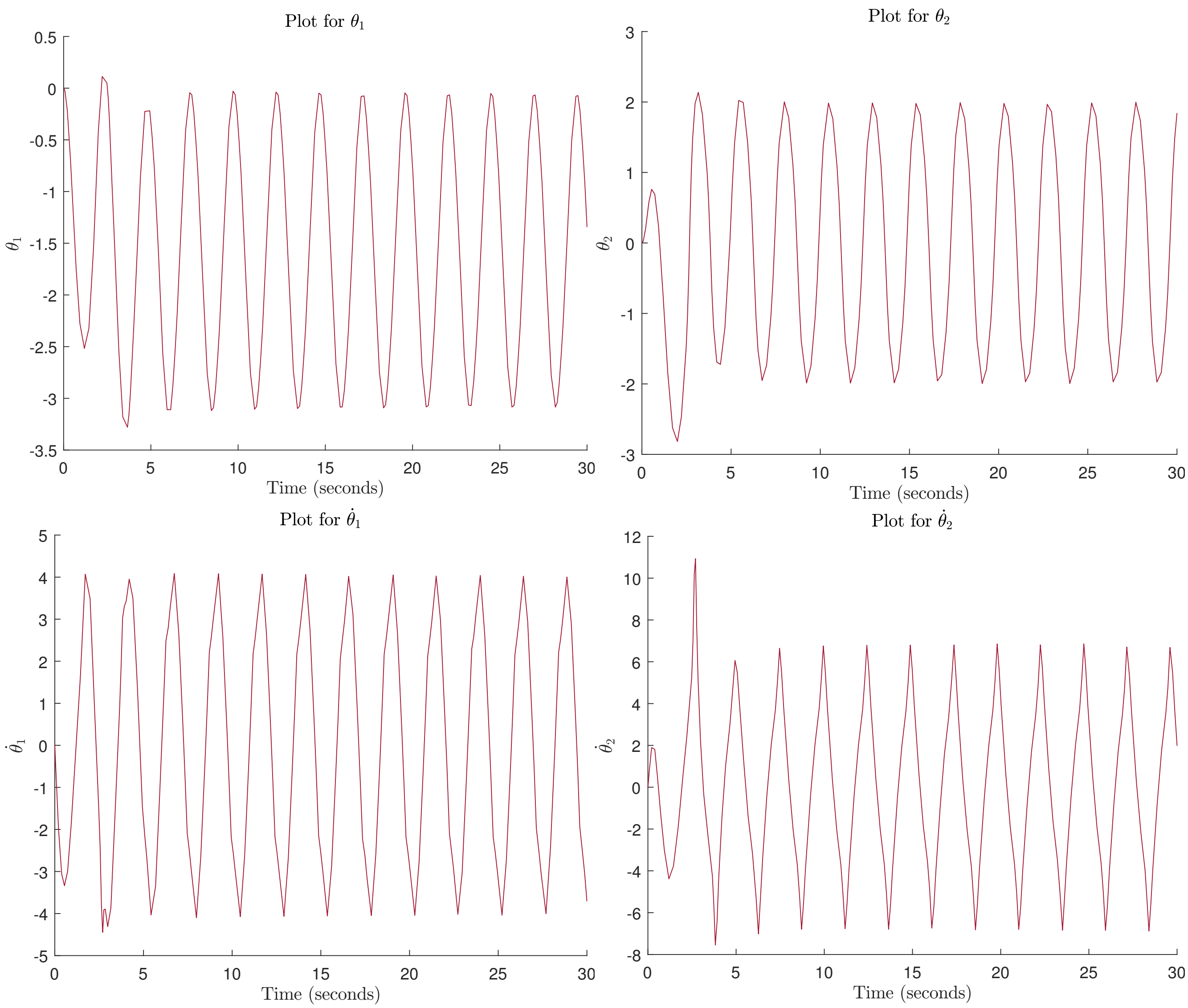

Figure 1: Joint angle and velocity plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = \tau_2 = 0$

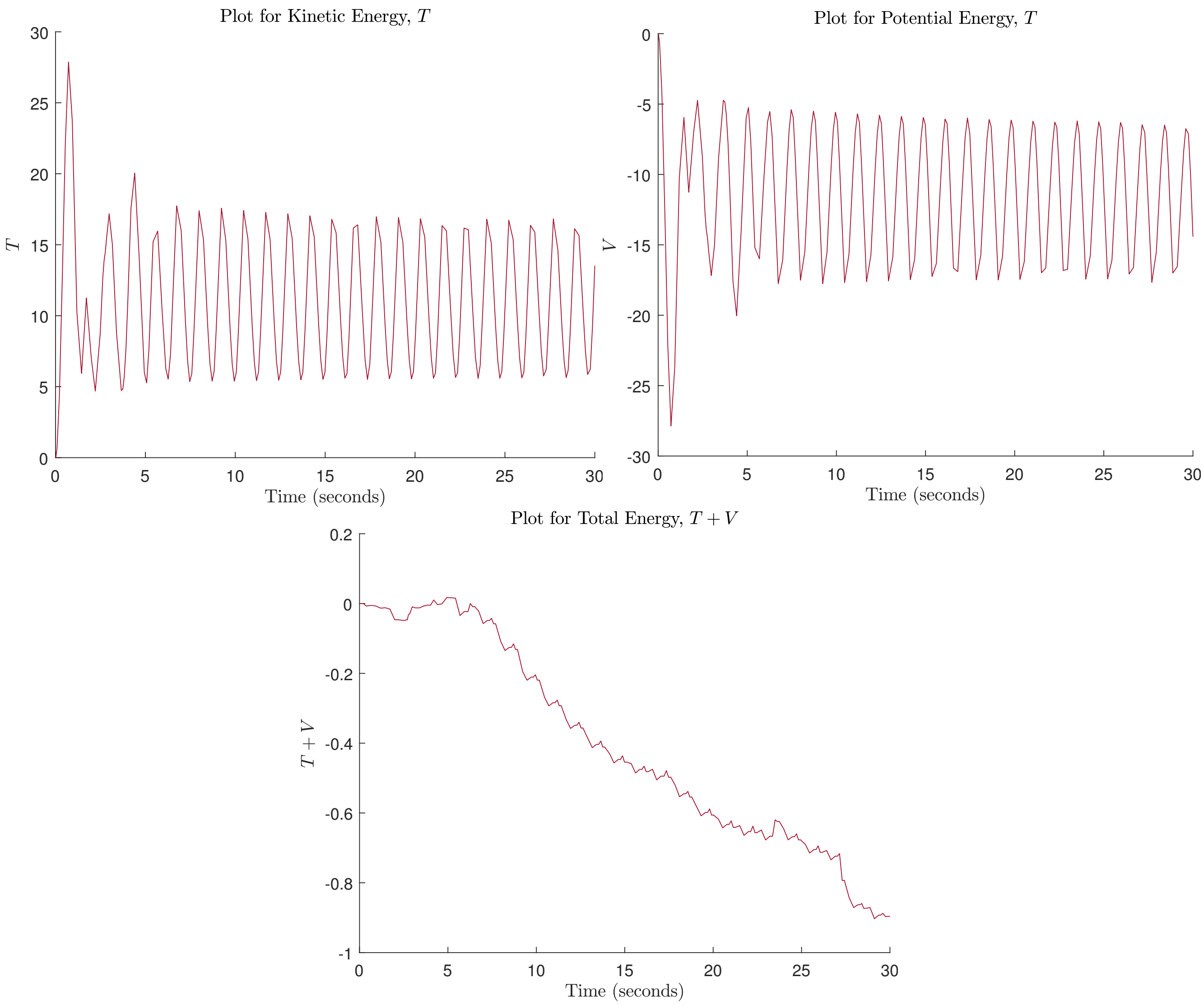

Figure 2: Energy plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = \tau_2 = 0$

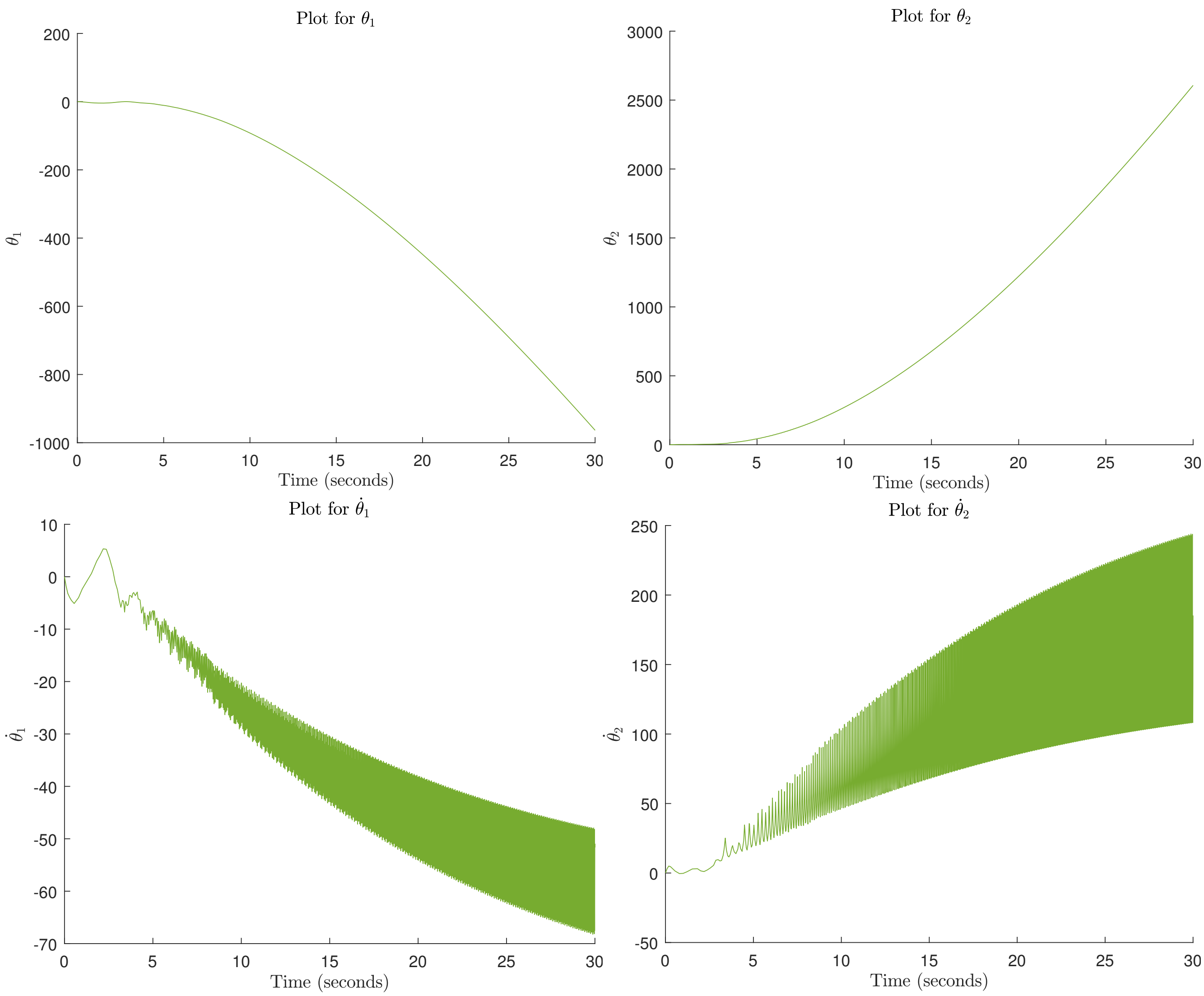

Figure 3: Joint angle and velocity plots for $x(0) = \begin{bmatrix} 0 & \frac{\pi}{2} & 0 & 0 \end{bmatrix}^T$, $\tau_1 = 0$, $\tau_2 = 5$

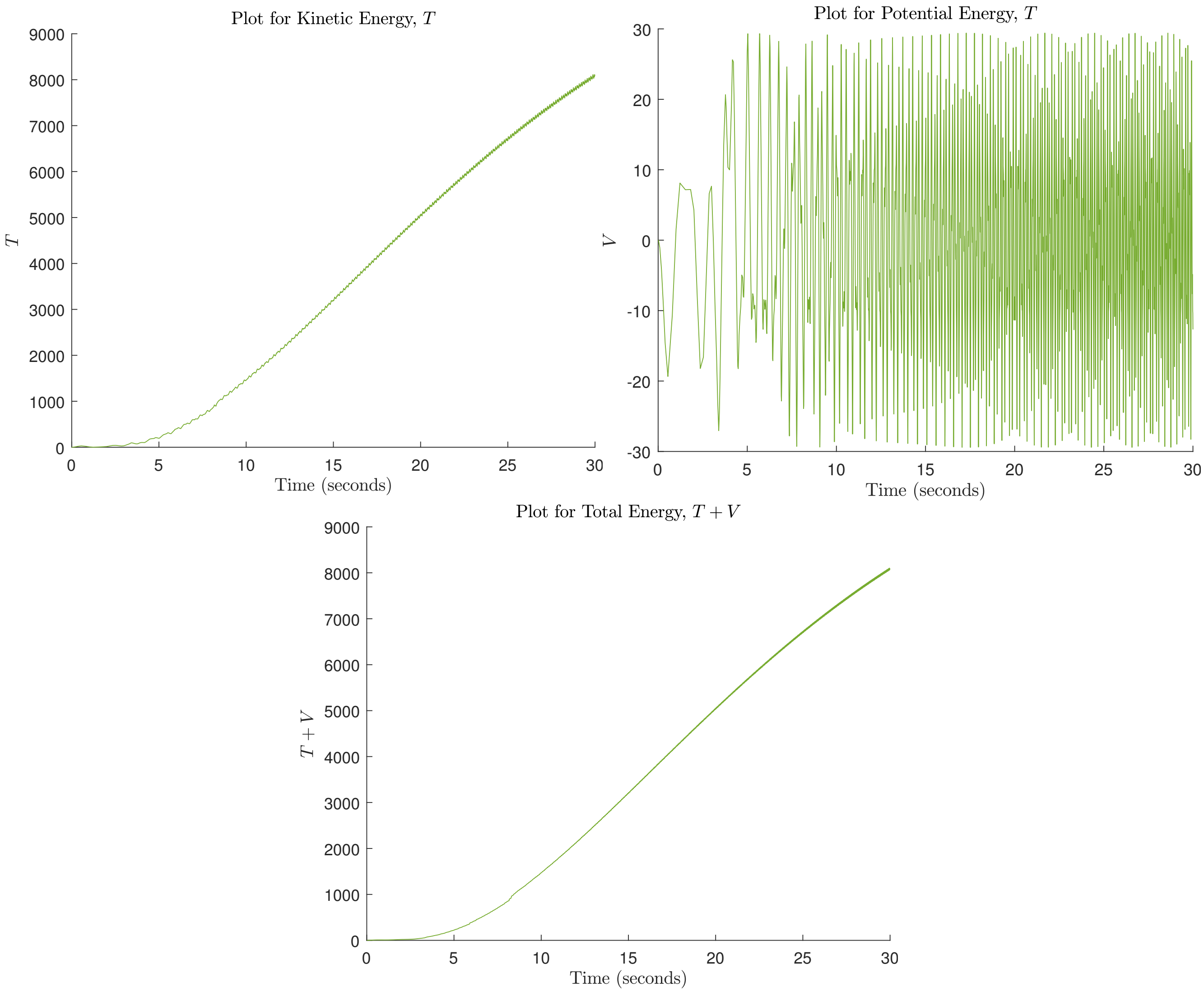

Figure 4: Energy plots for $x(0) = \begin{bmatrix} 0 & \frac{\pi}{2} & 0 & 0 \end{bmatrix}^T$, $\tau_1 = 0$, $\tau_2 = 5$

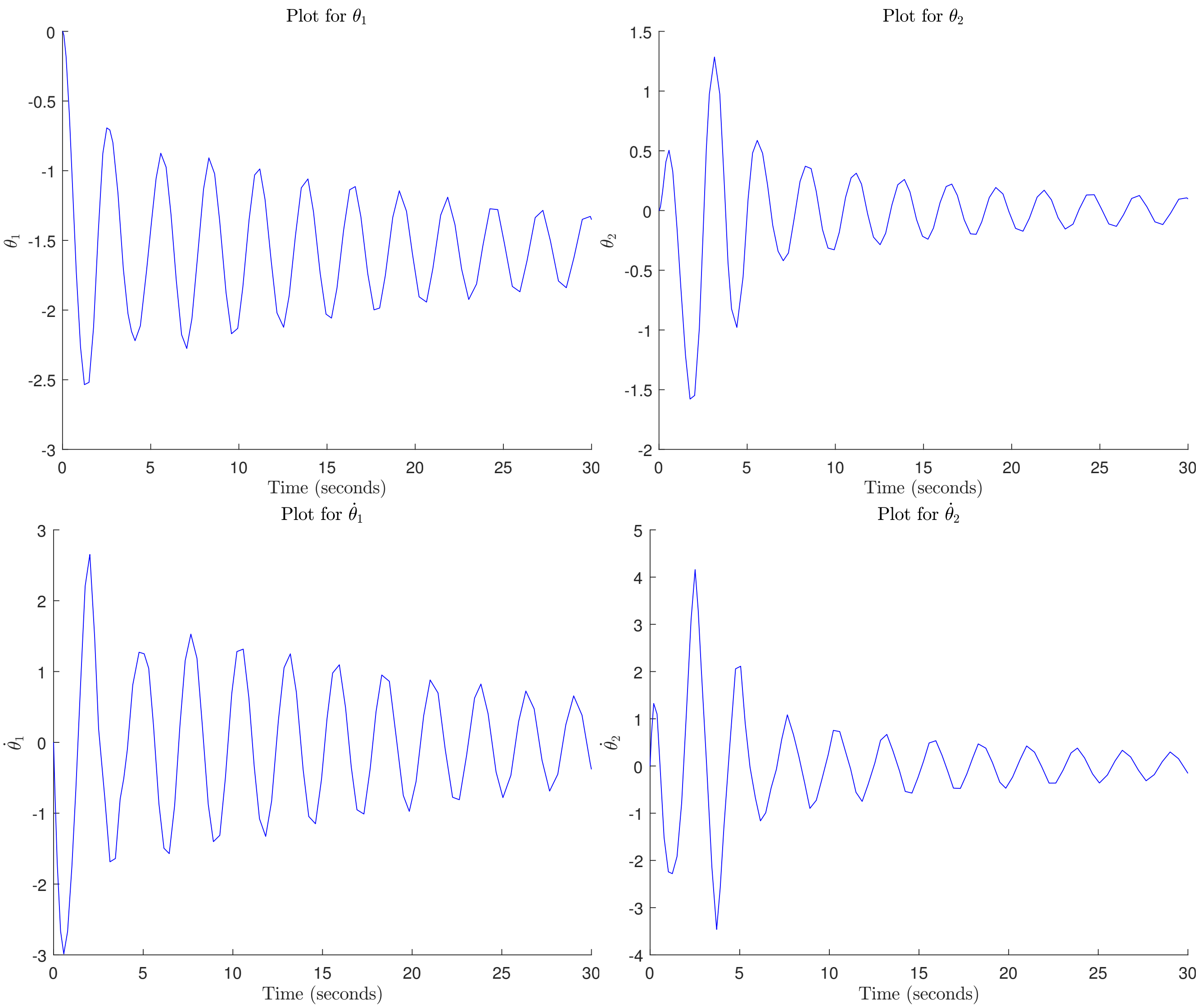

Figure 5: Joint angle and velocity plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = -0.5\dot{\theta}_1$, $\tau_2 = -0.5\dot{\theta}_2$

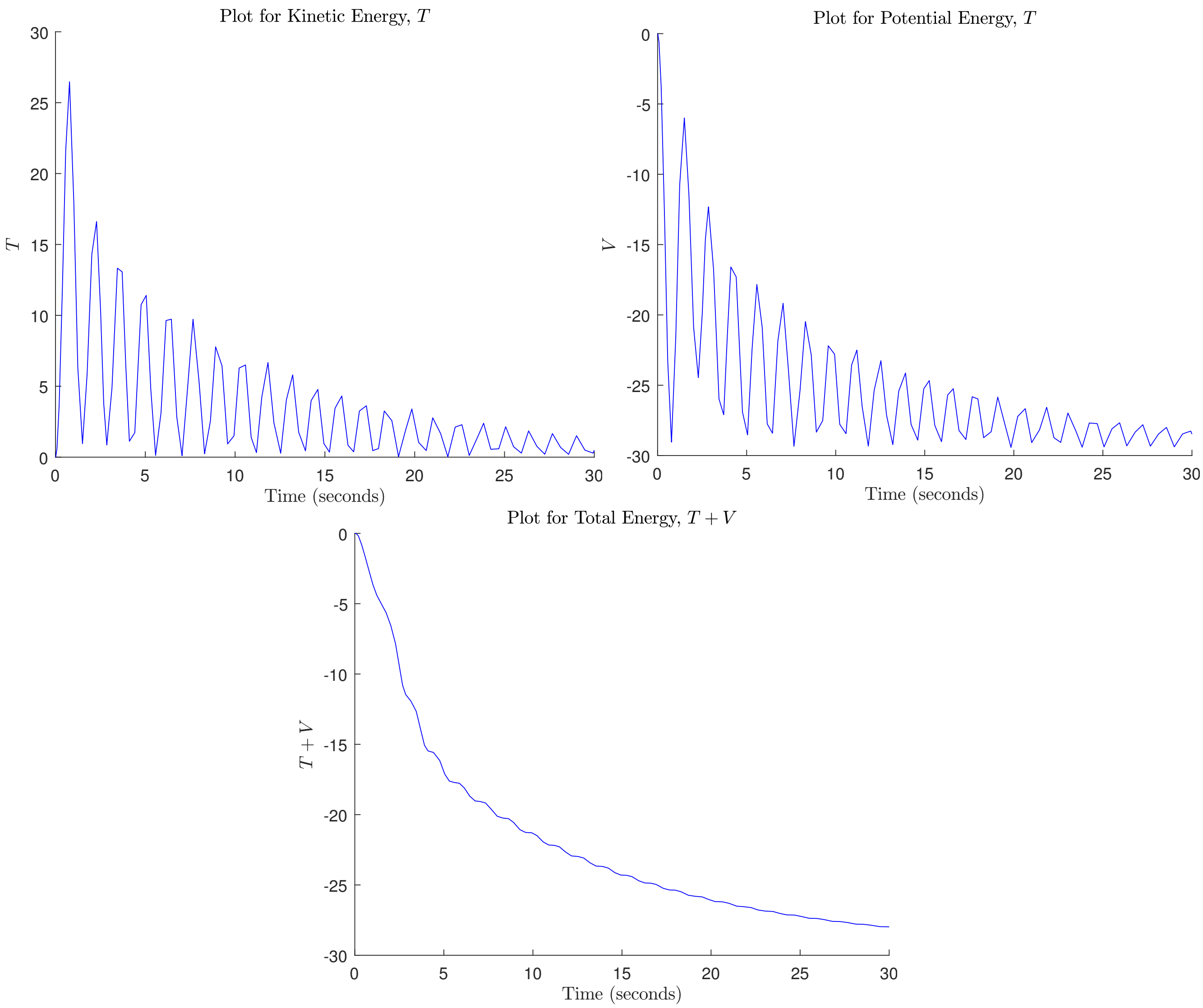

Figure 6: Energy plots for $x(0) = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$, $\tau_1 = -0.5\dot{\theta}_1$, $\tau_2 = -0.5\dot{\theta}_2$

1.1.3 Simulink block diagrams

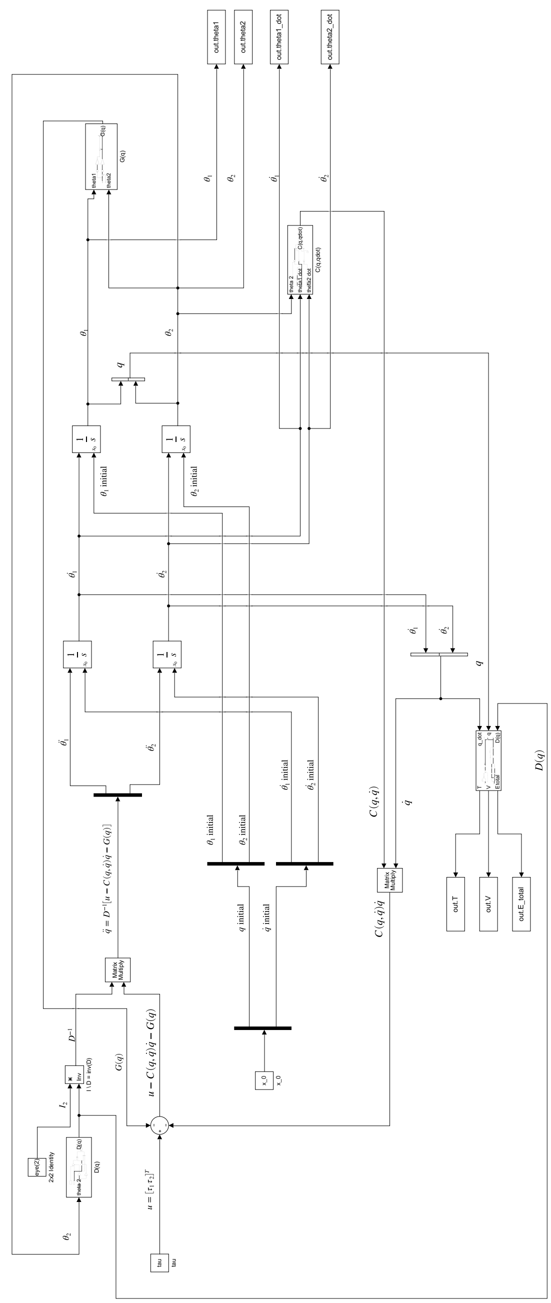

Figure 7: Simulink Block Diagram for the robot.

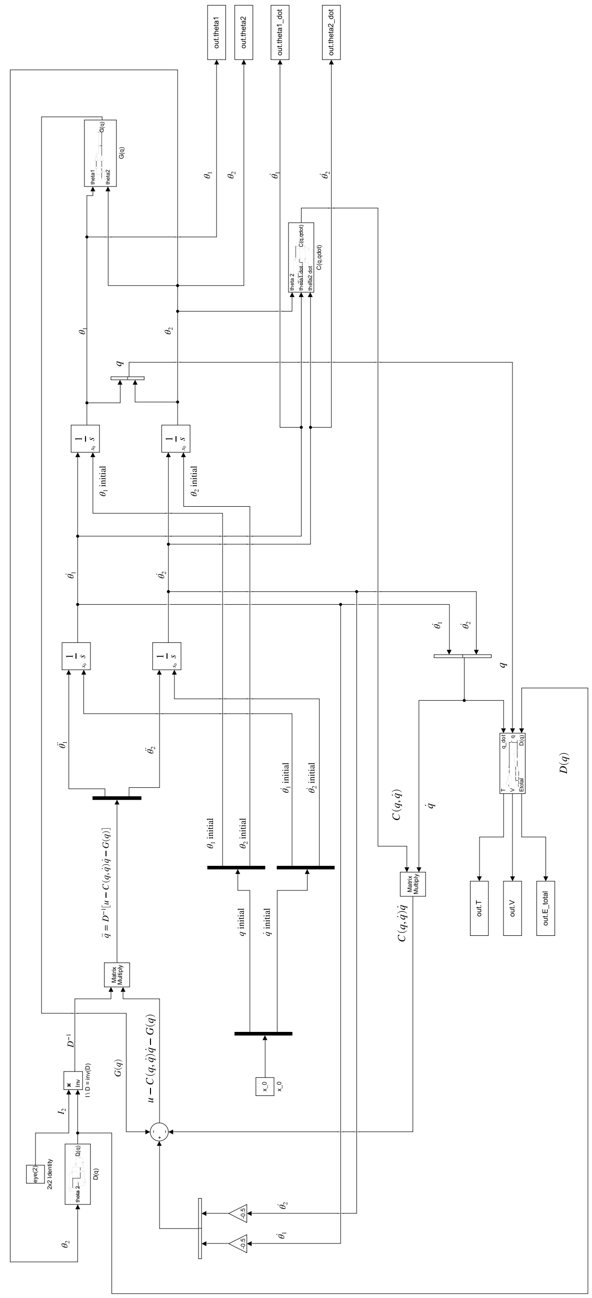

Figure 8: Simulink Block Diagram for the robot, used in part 1(c) to feed friction back into the system.)

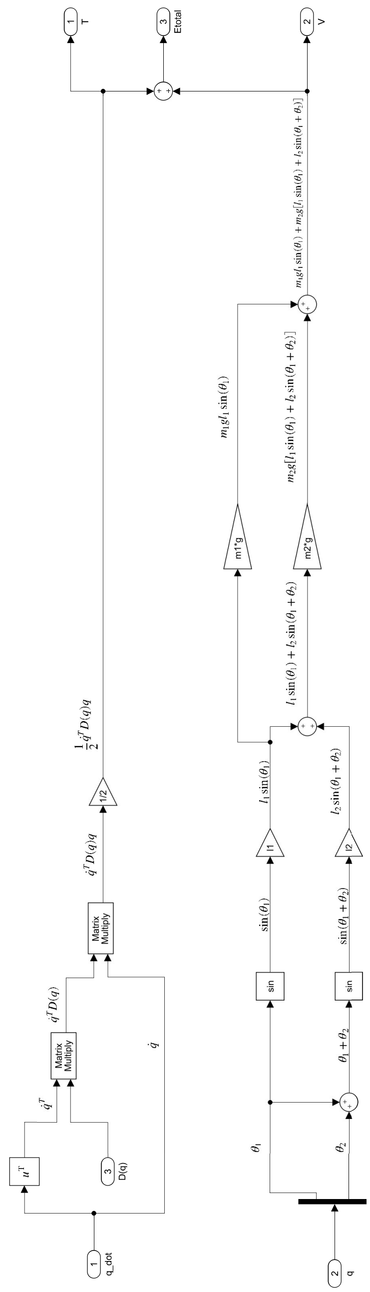

Figure 9: Simulink Block Diagram for the the subsystem used to calculate (potetial, kinetic, and total) energy.

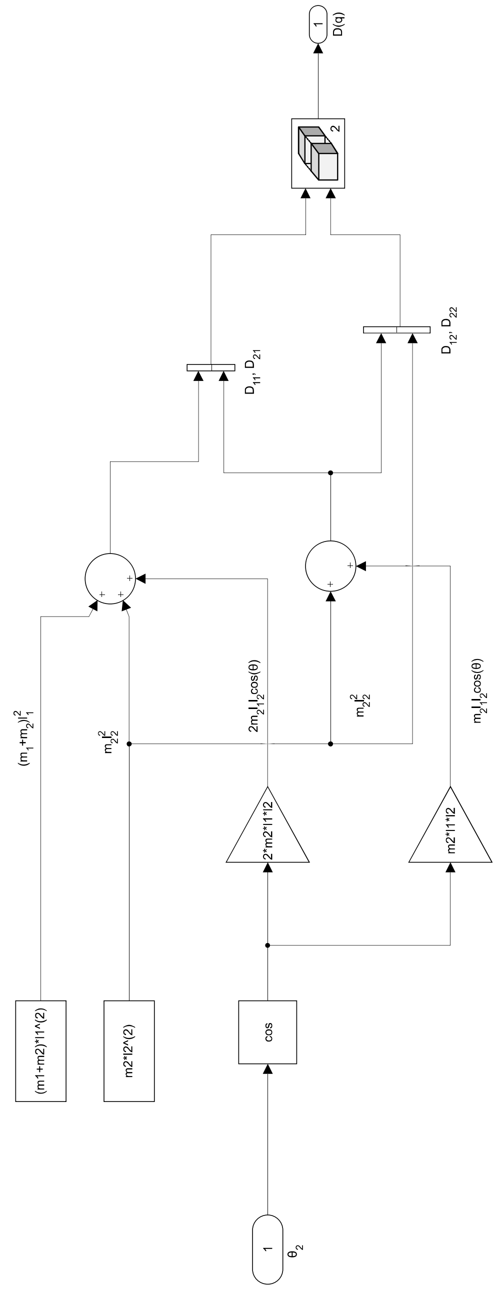

Figure 10: Simulink Block Diagram for the the subsystem used to calculate the $D(\boldsymbol{q})$ matrix.

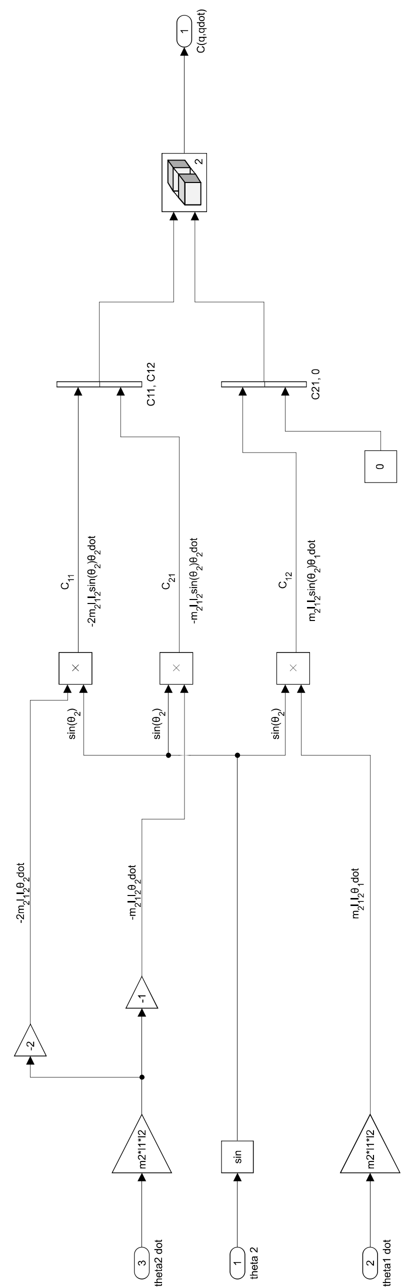

Figure 11: Simulink Block Diagram for the the subsystem used to calculate the $C(\boldsymbol{q}, \dot{\boldsymbol{q}})$ matrix.

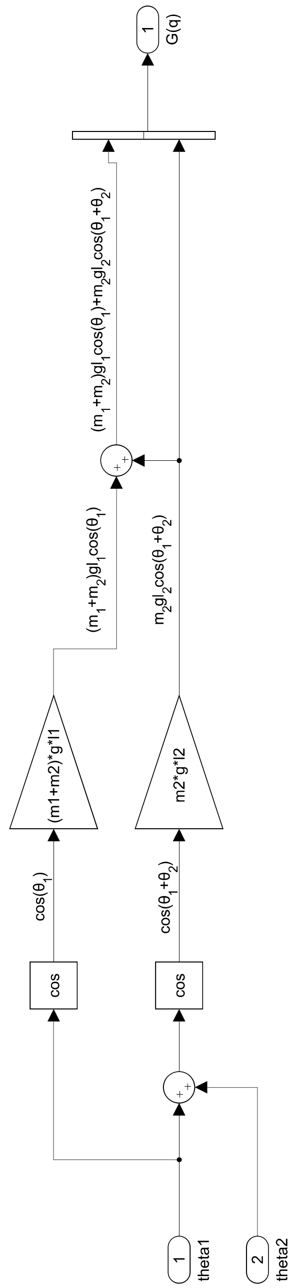

Figure 12: Simulink Block Diagram for the the subsystem used to calculate the $G(\boldsymbol{q})$ matrix.

2 Closed-Loop Controller Implementation

2.1 Joint Space Control

2.2 Task Space Control