Programming Assignment 3

Clear Environment

```
clc;
clear all;
close all;
warning ('off', 'all'); % Stopping all warnings
```

Loading Model Parameters

```
mass_1 = 1;
mass_2 = 1;
arm_1_length = 1;
arm_2_length = 1;
center_of_mass_distance_1 = 0.45;
center_of_mass_distance_2 = 0.45;
moment_of_inertia_1 = 0.084;
moment_of_inertia_2 = 0.084;
gravitational_acceleration = 9.81;
```

Part A: Trajectory Generation

```
syms a0 a1 a2 a3 t
theta_configuration = a0 + a1 * t + a2 * t^2 + a3 * t^3
```

theta_configuration = $a_0 + a_1\,t + a_2\,t^2 + a_3\,t^3$

```
theta_dot_configuration = diff(theta_configuration, t)
```

theta_dot_configuration = $a_1 + 2\,a_2\,t + 3\,a_3\,t^2$

```
timespan = [0 10];
theta_1_configuration = [deg2rad(180) deg2rad(0)];
theta_2_configuration = [deg2rad(90) deg2rad(0)];
theta_dot_1_configuration = [deg2rad(0) deg2rad(0)];
theta_dot_2_configuration = [deg2rad(0) deg2rad(0)];

% Calculating theta1 Trajectory
eq1 = subs(theta_configuration, t, timespan(1, 1)) == theta_1_configuration(1, 1);
eq2 = subs(theta_configuration, t, timespan(1, 2)) == theta_1_configuration(1, 2);
eq3 = subs(theta_dot_configuration, t, timespan(1, 1)) == theta_dot_1_configuration(1,
eq4 = subs(theta_dot_configuration, t, timespan(1, 2)) == theta_dot_1_configuration(1,
[q10, q11, q12, q13] = solve([eq1, eq2, eq3, eq4], [a0, a1, a2, a3]);
theta1_trajectory = q10 + q11 * t + q12 * t^2 + q13 * t^3;
theta1_dot_trajectory = diff(theta1_trajectory, t);
theta1_ddot_trajectory = diff(theta1_trajectory, t, 2);
x1d_trajectory = formula([theta1_trajectory; theta1_dot_trajectory])
```

```
x1d_trajectory =
```

$$\begin{bmatrix} \pi - \frac{3\pi t^2}{100} + \frac{\pi t^3}{500} \\ -\frac{3\pi t}{50} + \frac{3\pi t^2}{500} \end{bmatrix}$$

```
v1d_trajectory = theta1_ddot_trajectory
```

```
v1d_trajectory =
```

$$-\frac{3\pi}{50} + \frac{3\pi t}{250}$$

```
% Calculating theta2 Trajectory
eq1 = subs(theta_configuration, t, timespan(1, 1)) == theta_2_configuration(1, 1);
eq2 = subs(theta_configuration, t, timespan(1, 2)) == theta_2_configuration(1, 2);
eq3 = subs(theta_dot_configuration, t, timespan(1, 1)) == theta_dot_2_configuration(1,
eq4 = subs(theta_dot_configuration, t, timespan(1, 2)) == theta_dot_2_configuration(1,
[q20, q21, q22, q23] = solve([eq1, eq2, eq3, eq4], [a0, a1, a2, a3]);
theta2_trajectory = q20 + q21 * t + q22 * t^2 + q23 * t^3;
theta2_dot_trajectory = diff(theta2_trajectory, t);
theta2_ddot_trajectory = diff(theta2_trajectory, t, 2);
x2d_trajectory = formula([theta2_trajectory; theta2_dot_trajectory])
```

```
x2d_trajectory =
```

$$\begin{bmatrix} \frac{\pi}{2} - \frac{3\pi t^2}{200} + \frac{\pi t^3}{1000} \\ -\frac{3\pi t}{100} + \frac{3\pi t^2}{1000} \end{bmatrix}$$

```
v2d_trajectory = theta2_ddot_trajectory
```

```
v2d_trajectory =
```

$$-\frac{3\pi}{100} + \frac{3\pi t}{500}$$

Part B: Manipulator Form Derivation from Programming Assignment 1

```
prog_assignment1;
ddq = simplify(ddq);
syms ddq1 ddq2 dq1 dq2 q1 q2
ddq_mf = subs(ddq, [m1, m2, r1, r2, I1, I2, l1, l2, g], [mass_1, mass_2, center_of_mass
ddq_mf = subs(ddq_mf, [diff(theta1, t, 2), diff(theta2, t, 2)], [ddq1, ddq2]);
ddq_mf = subs(ddq_mf, [diff(theta1, t), diff(theta2, t)], [dq1, dq2]);
ddq_mf = subs(ddq_mf, [theta1, theta2], [q1, q2]);
[tau1, tau2] = solve(ddq_mf, [T1,T2]);
tau = formula(simplify([tau1;tau2]))
```

```
tau =
```

$$\left[\begin{matrix} \frac{1573\,\mathrm{ddq}_1}{1000}+\frac{573\,\mathrm{ddq}_2}{2000}-\frac{8829\sin(q_2+q_1)}{2000}-\frac{28449\sin(q_1)}{2000}-\frac{9\,\mathrm{dq}_2{}^2\sin(q_2)}{20}+\frac{9\,\mathrm{ddq}_1\cos(q_2)}{10}+\frac{9\,\mathrm{ddq}_2\cos(\ldots}{20} \\ \frac{573\,\mathrm{ddq}_1}{2000}+\frac{573\,\mathrm{ddq}_2}{2000}-\frac{8829\sin(q_2+q_1)}{2000}+\frac{9\,\mathrm{dq}_1{}^2\sin(q_2)}{20}+\frac{9\,\mathrm{ddq}_1\cos(q_2)}{20} \end{matrix}\right.$$

```
clear m1 m2 r1 r2 I1 I2 l1 l2 g;

%Gravity Matrix
gravity_matrix = formula(simplify(subs(tau, [ddq1, dq1, ddq2, dq2], [0, 0, 0, 0])))
```

gravity_matrix =

$$\begin{bmatrix} -\frac{8829\sin(q_2+q_1)}{2000}-\frac{28449\sin(q_1)}{2000} \\ -\frac{8829\sin(q_2+q_1)}{2000} \end{bmatrix}$$

```
%Mass Matrix
mass_matrix = formula(simplify(subs(tau, [dq1, dq2], [0, 0]) - gravity_matrix));
mass_matrix = collect(mass_matrix,[ddq1,ddq2]);
mass_matrix = formula([subs(mass_matrix(1,:),[ddq1,ddq2],[1,0]),subs(mass_matrix(1,:),[
```

mass_matrix =

$$\begin{bmatrix} \frac{9\cos(q_2)}{10}+\frac{1573}{1000} & \frac{9\cos(q_2)}{20}+\frac{573}{2000} \\ \frac{9\cos(q_2)}{20}+\frac{573}{2000} & \frac{573}{2000} \end{bmatrix}$$

```
%Coriolis Matrix
coriolis_matrix = formula(simplify(tau - gravity_matrix - mass_matrix*[ddq1;ddq2]))
```

coriolis_matrix =

$$\begin{bmatrix} -\frac{9\,\mathrm{dq}_2\sin(q_2)\ (\mathrm{dq}_2+2\,\mathrm{dq}_1)}{20} \\ \frac{9\,\mathrm{dq}_1{}^2\sin(q_2)}{20} \end{bmatrix}$$

```
system_input = formula(mass_matrix*[ddq1;ddq2] + coriolis_matrix + gravity_matrix);
```

Part C: Feedback Linearlization

```
%For diff(theta1,t,2)
%v1 = ddq1;
x11 = q1;
x12 = dq1;
x1 = [x11; x12];
%dx1 = [x12; v1];
A1 = [0 1; 0 0];
B1 = [0; 1];
%A_sys = A - BK
```

```
lambda1 = [-3, -4]
```

```
lambda1 = 1×2
    -3    -4
```

```
K1 = place(A1, B1, lambda1)
```

```
K1 = 1×2
   12.0000    7.0000
```

```
v1 = (-K1 * (x1 - x1d_trajectory)) + v1d_trajectory;

% For diff(theta2,t,2)
%v2 = ddq2;
x21 = q2;
x22 = dq2;
x2 = [x21; x22];
%dx2 = [x22; v2];
A2 = [0 1; 0 0];
B2 = [0; 1];
%A_sys = A - BK
lambda2 = [-5, -6]
```

```
lambda2 = 1×2
    -5    -6
```

```
K2 = place(A2, B2, lambda2)
```

```
K2 = 1×2
   30.0000   11.0000
```

```
v2 = (-K2 * (x2 - x2d_trajectory)) + v2d_trajectory;

% Virtual Control Input
v = formula([v1; v2])
```

v =

$$\begin{bmatrix} \frac{597\,\pi}{50} - 12\,q_1 - 7\,\mathrm{dq}_1 - \frac{51\,\pi\,t}{125} - \frac{159\,\pi\,t^2}{500} + \frac{3\,\pi\,t^3}{125} \\ \frac{1497\,\pi}{100} - 30\,q_2 - 11\,\mathrm{dq}_2 - \frac{81\,\pi\,t}{250} - \frac{417\,\pi\,t^2}{1000} + \frac{3\,\pi\,t^3}{100} \end{bmatrix}$$

```


% State FeedBack Control Strategy
system_input = subs(system_input, [ddq1, ddq2], [v(1), v(2)])
```

system_input =

$$\Bigg[-\frac{8829\sin(q_2+q_1)}{2000} - \frac{28449\sin(q_1)}{2000} - \left(\frac{9\cos(q_2)}{10} + \frac{1573}{1000}\right)\left(-\frac{597\,\pi}{50} + 12\,q_1 + 7\,\mathrm{dq}_1 + \frac{51\,\pi\,t}{125} + \frac{159\,\pi\,t^2}{500}\right.$$
$$\frac{857781\,\pi}{200000} - \frac{1719\,q_2}{200} - \frac{6303\,\mathrm{dq}_2}{2000} - \frac{8829\sin(q_2+q_1)}{2000} + \frac{9\,\mathrm{dq}_1{}^2\sin(q_2)}{20} - \frac{46413\,\pi\,t}{500000} - \frac{23}{2}$$

Part D: Update ODE Function from Programming Assignment 2

```
syms x1(t) x2(t) x3(t) x4(t)
%x = [x1;x2;x3;x4] = [theta1; dtheta1; theta2; dtheta2]
%dx = [dx1;dx2;dx3;dx4] = [dtheta1; ddtheta1; dtheta2; ddtheta2]
%dx = [x2;ddq1;x3;ddq2]
%dx = subs(ddq_mf,[T1,T2],[system_input(1),system_input(2)]);
%[mddq1,mddq2] = solve(dx,[ddq1,ddq2]);
%v_func = [x2; mddq1; x4; mddq2];
v_func = [x2; v(1); x4; v(2)];
v_func = subs(v_func, [q1, dq1, q2, dq2], [x1, x2, x3, x4]);
vars = [x1(t) x2(t) x3(t) x4(t)];
% Modified prog_assignment2.m
```

Part E: Simulate the system using ODE45

```
initial_params = [deg2rad(200); deg2rad(0); deg2rad(125); deg2rad(0)];
%clear time state_space_matrix system_plot_points theta_plot_points;
prog_assignment2;
t_size = size(time);
t_size = t_size(1, 1);
theta_plot_points = zeros(t_size, 4);
system_input_points = zeros(t_size, 2);

for i = 1:t_size
    theta_plot_points(i, 1) = (simplify(subs(x1d_trajectory(1), t, time(i, 1))));
    theta_plot_points(i, 2) = (simplify(subs(x1d_trajectory(2), t, time(i, 1))));
    theta_plot_points(i, 3) = (simplify(subs(x2d_trajectory(1), t, time(i, 1))));
    theta_plot_points(i, 4) = (simplify(subs(x2d_trajectory(2), t, time(i, 1))));
    system_input_points(i, 1) = double(simplify(subs(system_input(1), [q1, dq1, q2, dq2
    system_input_points(i, 2) = double(simplify(subs(system_input(2), [q1, dq1, q2, dq2
    virtual_input(i,1) = double(simplify(subs(v1, [q1, dq1, q2, dq2, t], [state_space_m
    virtual_input(i,2) = double(simplify(subs(v2, [q1, dq1, q2, dq2, t], [state_space_m
end

tau1_max = max(system_input_points(:, 1))
```

```
tau1_max = 6.8820
```

```
tau1_min = min(system_input_points(:, 1))
```

```
tau1_min = -17.8803
```

```
tau2_max = max(system_input_points(:, 2))
```

```
tau2_max = 4.9751
```

```
tau2_min = min(system_input_points(:, 2))
```

```
tau2_min = -4.3339
```

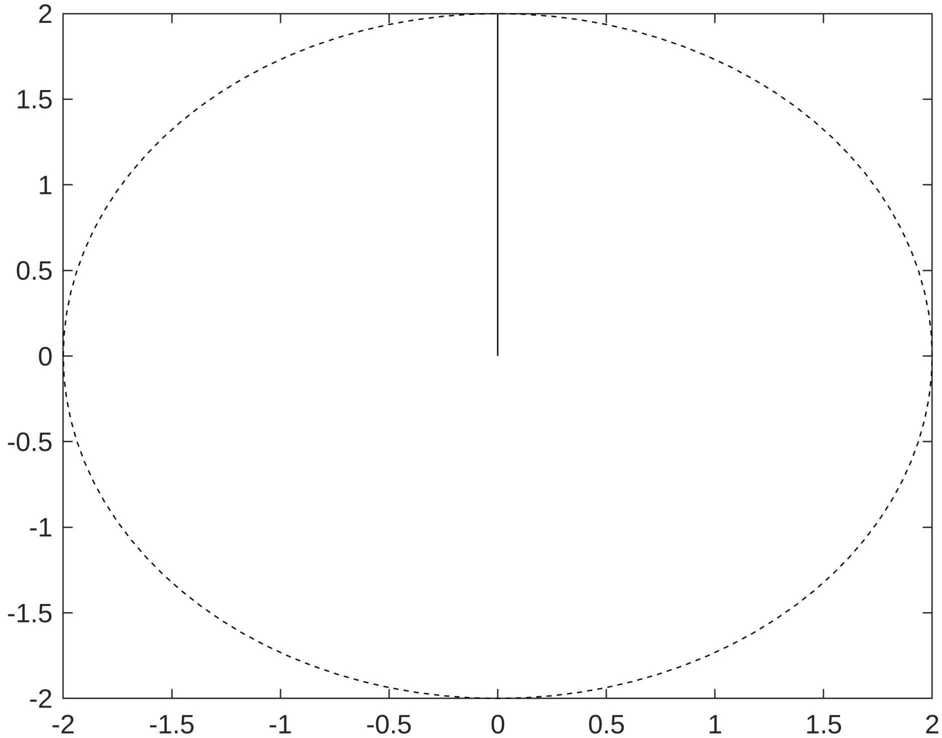
Calculating Link Positions

```
actual_position = zeros(t_size(1, 1), 4);
trajectory_position = zeros(t_size(1, 1), 2);

for i = 1:t_size(1, 1)
    actual_position(i, 1) = arm_1_length * sin(state_space_matrix(i, 1));
    actual_position(i, 2) = arm_1_length * cos(state_space_matrix(i, 1));
    actual_position(i, 3) = actual_position(i, 1) + arm_2_length * sin(state_space_matr
    actual_position(i, 4) = actual_position(i, 2) + arm_2_length * cos(state_space_matr
    trajectory_position(i, 1) = arm_1_length * sin(theta_plot_points(i, 1));
    trajectory_position(i, 2) = arm_1_length * cos(theta_plot_points(i, 1));
    trajectory_position(i, 3) = trajectory_position(i, 1) + arm_2_length * sin(theta_pl
    trajectory_position(i, 4) = trajectory_position(i, 2) + arm_2_length * cos(theta_pl
end
```

Animation

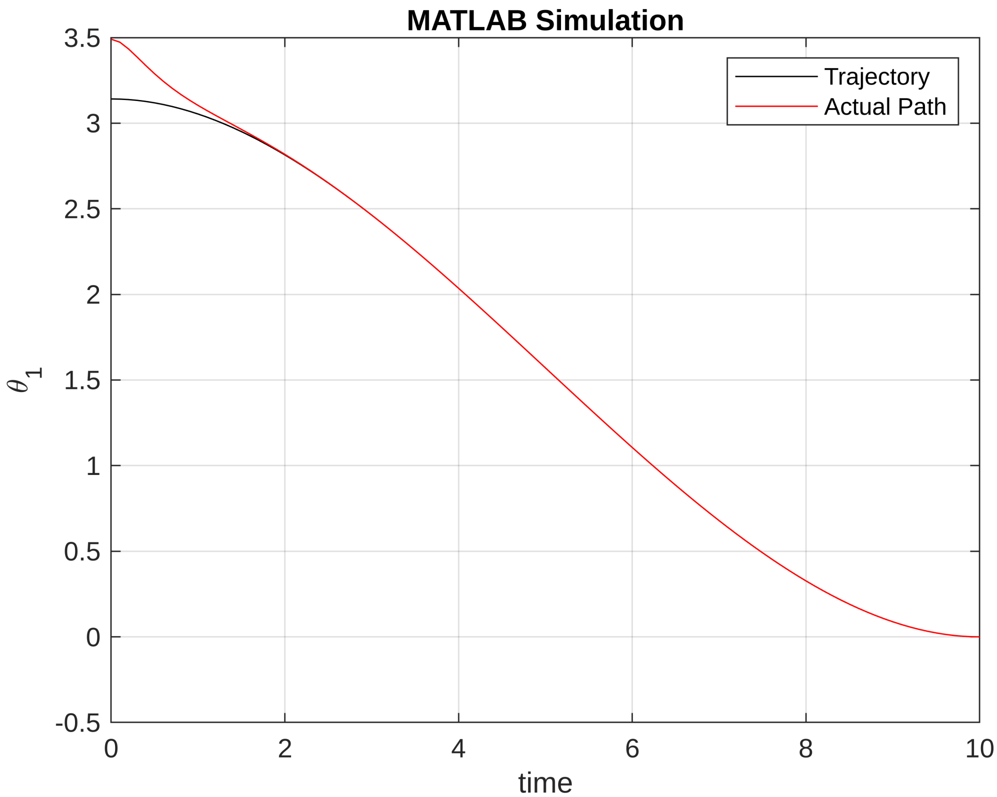
```
xAxisArrayXCoordinates = [-2 2];
xAxisArrayYCoordinates = [0 0];
yAxisArrayXCoordinates = [0 0];
yAxisArrayYCoordinates = [-2 2];
th = 0:pi / 50:2 * pi;
xunit = ((arm_1_length + arm_2_length) * cos(th));
yunit = ((arm_1_length + arm_2_length) * sin(th));
animation = VideoWriter('MATLAB Simulation.avi');
open(animation);
for i = 1:1:t_size(1, 1)
    %Plotting Graph
    actuallink1XCoordinates = [0 actual_position(i, 1)];
    actuallink1YCoordinates = [0 actual_position(i, 2)];
    actuallink2XCoordinates = [actual_position(i, 1) actual_position(i, 3)];
    actuallink2YCoordinates = [actual_position(i, 2) actual_position(i, 4)];
    trajectorylink1XCoordinates = [0 trajectory_position(i, 1)];
    trajectorylink1YCoordinates = [0 trajectory_position(i, 2)];
    trajectorylink2XCoordinates = [trajectory_position(i, 1) trajectory_position(i, 3)]
    trajectorylink2YCoordinates = [trajectory_position(i, 2) trajectory_position(i, 4)]
    plot(xunit, yunit, 'k', 'LineStyle', '--'); % Draw Circular Axes
    hold on;
    plot(actuallink1XCoordinates, actuallink1YCoordinates, 'red');
    plot(actuallink2XCoordinates, actuallink2YCoordinates, 'blue');
    plot(trajectorylink1XCoordinates, trajectorylink1YCoordinates, 'black');
    plot(trajectorylink2XCoordinates, trajectorylink2YCoordinates, 'black');
    frame = getframe(gcf);
    writeVideo(animation,frame);
    pause(0.1); % pause to see realtime animation. Given in seconds
    hold off;
end
```



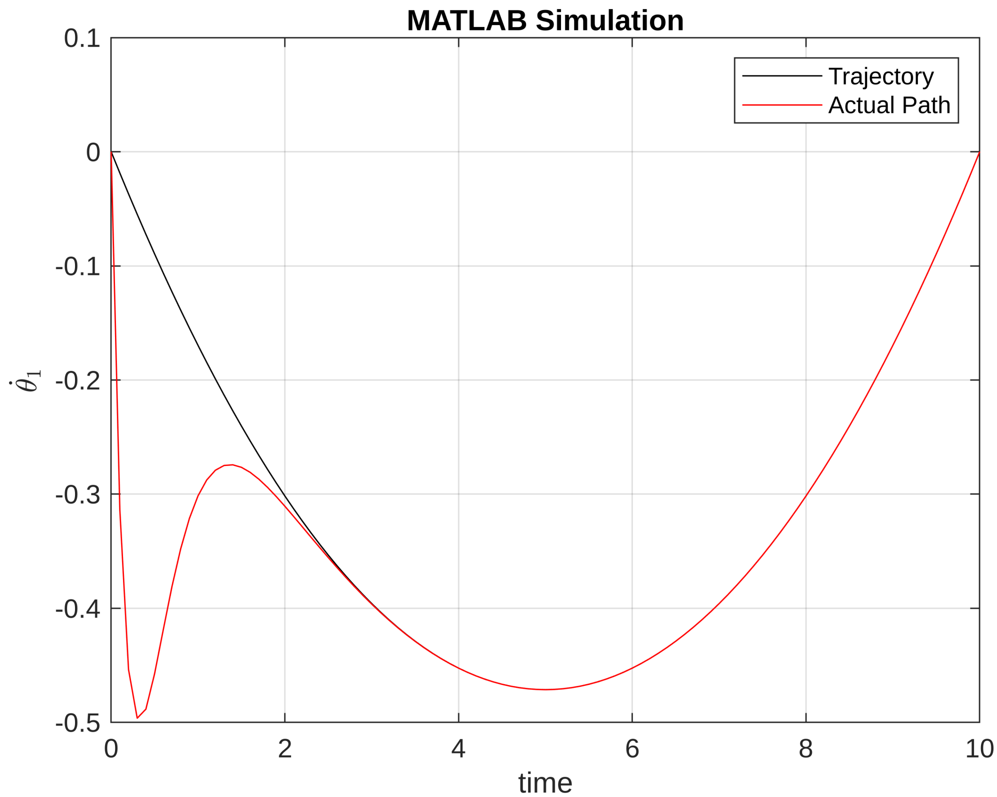

```
close(animation);
```

Part E (a): Plot Trajectories

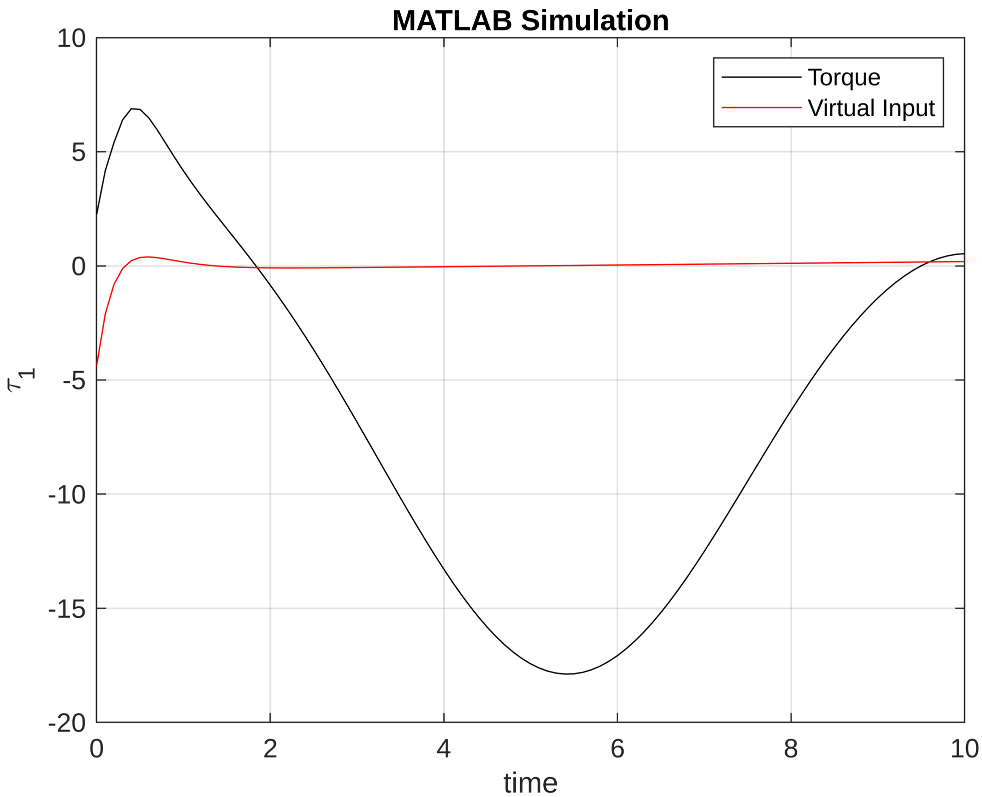
```
plot(time, theta_plot_points(:, 1), 'black')
hold on;
plot(time, state_space_matrix(:, 1), 'red')
legend('Trajectory', 'Actual Path')
hold off;
grid on;
xlabel('time');
ylabel('\theta_{1}')
title('MATLAB Simulation')
saveas(gcf, 'matlab_theta1.jpg')
```



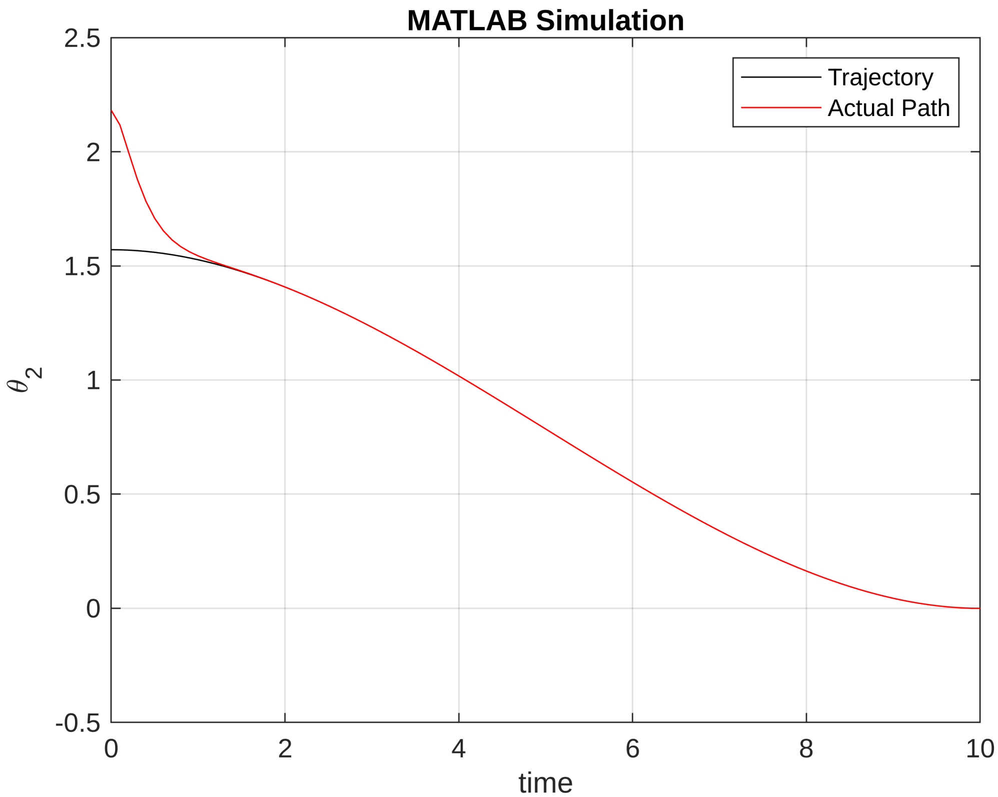

```
plot(time, theta_plot_points(:, 2), 'black')
hold on;
plot(time, state_space_matrix(:, 2), 'red')
legend('Trajectory', 'Actual Path')
hold off;
grid on;
xlabel('time');
ylabel('$\dot{\theta}_{1}$', 'Interpreter', 'latex')
title('MATLAB Simulation')
saveas(gcf, 'matlab_dtheta1.jpg')
```



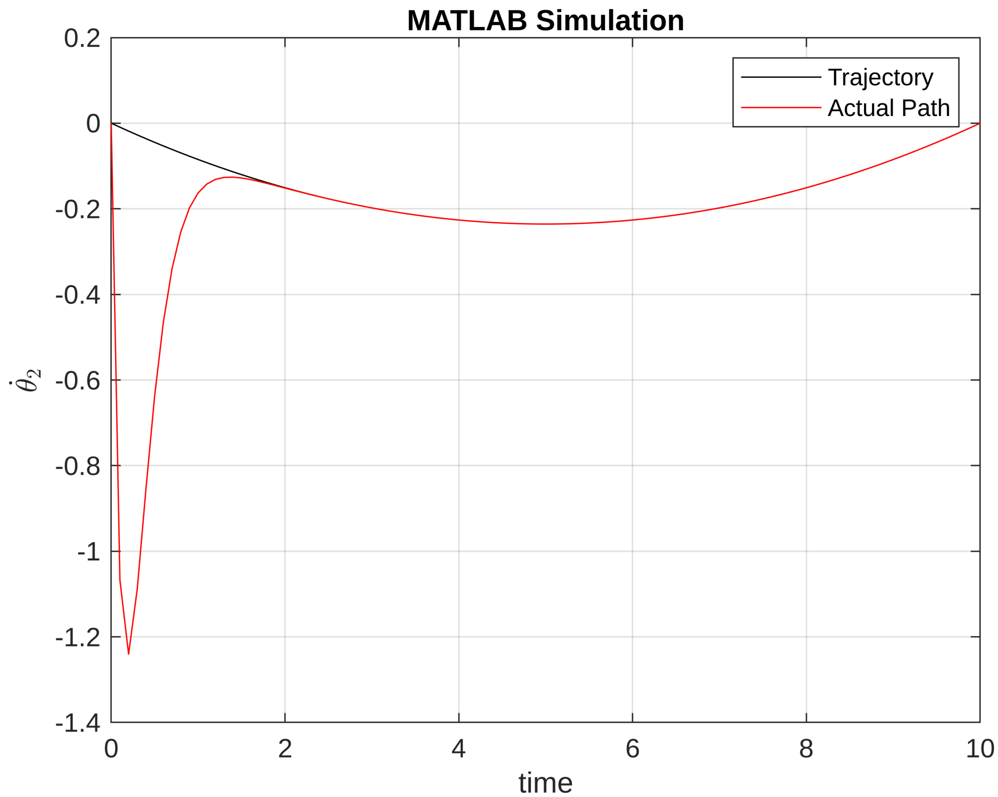

```
plot(time, system_input_points(:, 1), 'black')
hold on;
plot(time, virtual_input(:, 1), 'red')
hold off;
legend('Torque','Virtual Input')
grid on;
xlabel('time');
ylabel('\tau_{1}')
title('MATLAB Simulation')
saveas(gcf, 'matlab_tau1.jpg')
```



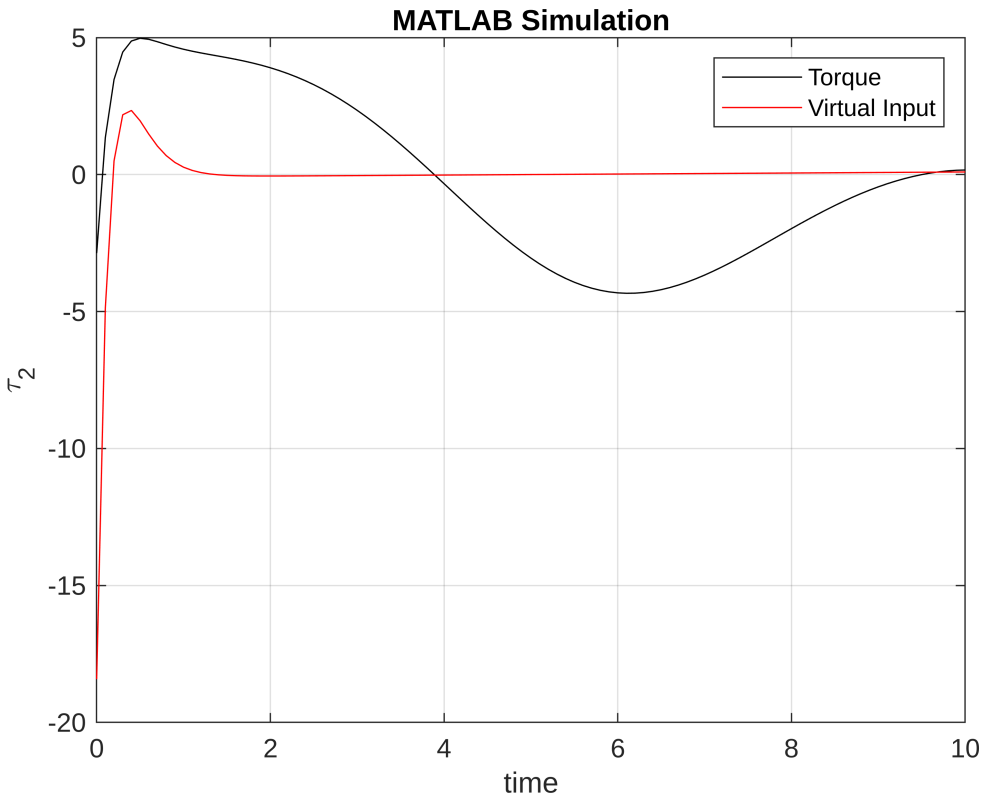

```
plot(time, theta_plot_points(:, 3), 'black')
hold on;
plot(time, state_space_matrix(:, 3), 'red')
legend('Trajectory', 'Actual Path')
hold off;
grid on;
xlabel('time');
ylabel('\theta_{2}')
title('MATLAB Simulation')
saveas(gcf, 'matlab_theta2.jpg')
```




```
plot(time, theta_plot_points(:, 4), 'black')
hold on;
plot(time, state_space_matrix(:, 4), 'red')
legend('Trajectory', 'Actual Path')
hold off;
grid on;
xlabel('time');
ylabel('$\dot{\theta}_{2}$', 'Interpreter', 'latex')
title('MATLAB Simulation')
saveas(gcf, 'matlab_dtheta2.jpg')
```




```
plot(time, system_input_points(:, 2), 'black')
hold on;
plot(time, virtual_input(:, 2), 'red')
hold off;
legend('Torque','Virtual Input')
grid on;
xlabel('time');
ylabel('\tau_{2}')
title('MATLAB Simulation')
saveas(gcf, 'matlab_tau2.jpg')
```



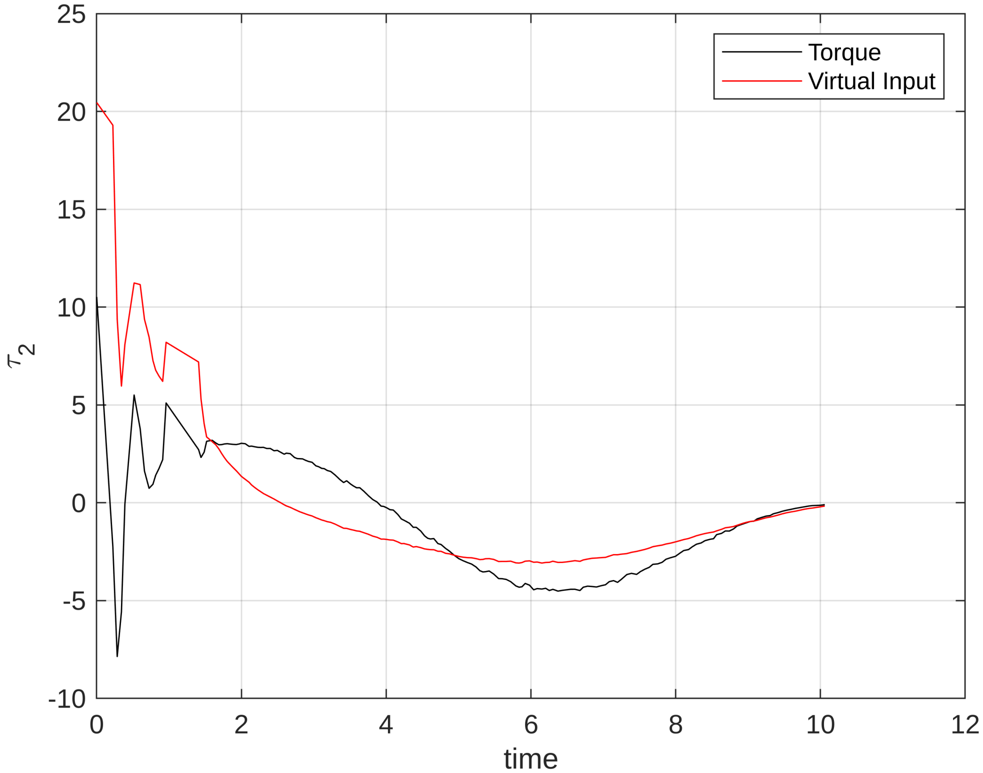
Saving Data

```
writematrix(system_input_points,'matlab_efforts.xls');
writematrix(state_space_matrix,'matlab_theta_plot_points.xls');
writematrix(virtual_input,'matlab_virtual_input.xls');
writematrix(time,'matlab_time.xls');
writematrix(theta_plot_points,'trajectory.xls');
```

Part F: Gazebo Simulation

```
clc;
clear all;
close all;
rrbot_traj_control% Run the file for next part
```

```
The value of the ROS_MASTER_URI environment variable, http://localhost:11311, will be used to connect to t
Initializing global node /matlab_global_node_60134 with NodeURI http://ubuntu-20:40315/ and MasterURI http
```



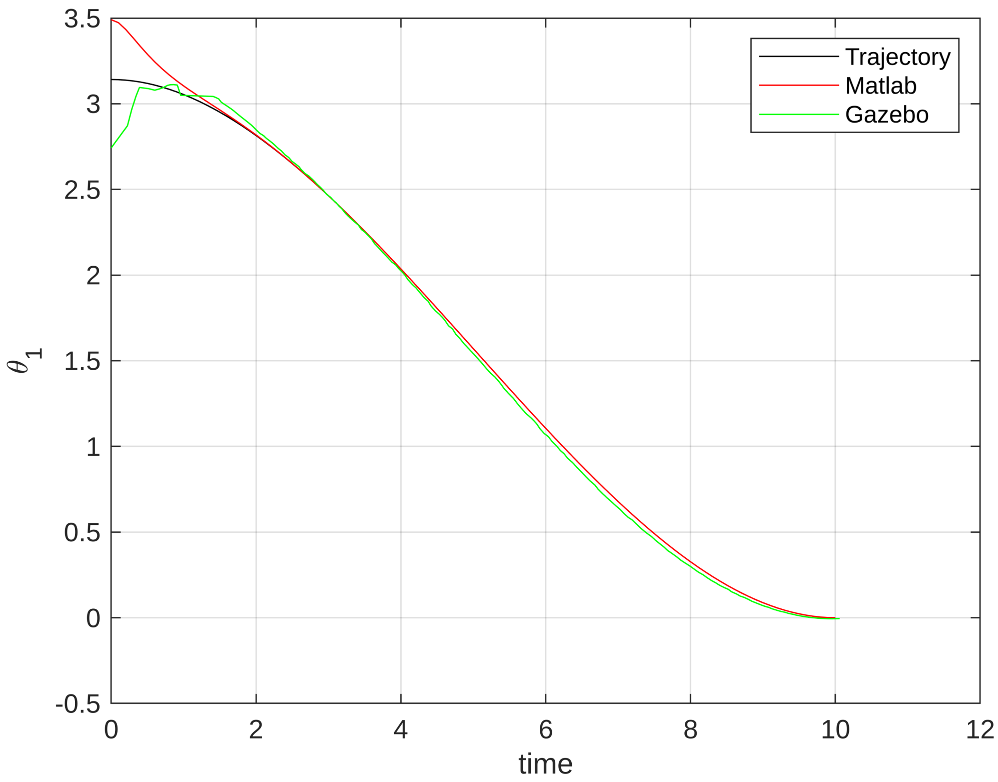

```
Shutting down global node /matlab_global_node_60134 with NodeURI http://ubuntu-20:40315/ and MasterURI htt
```

Comparing Results

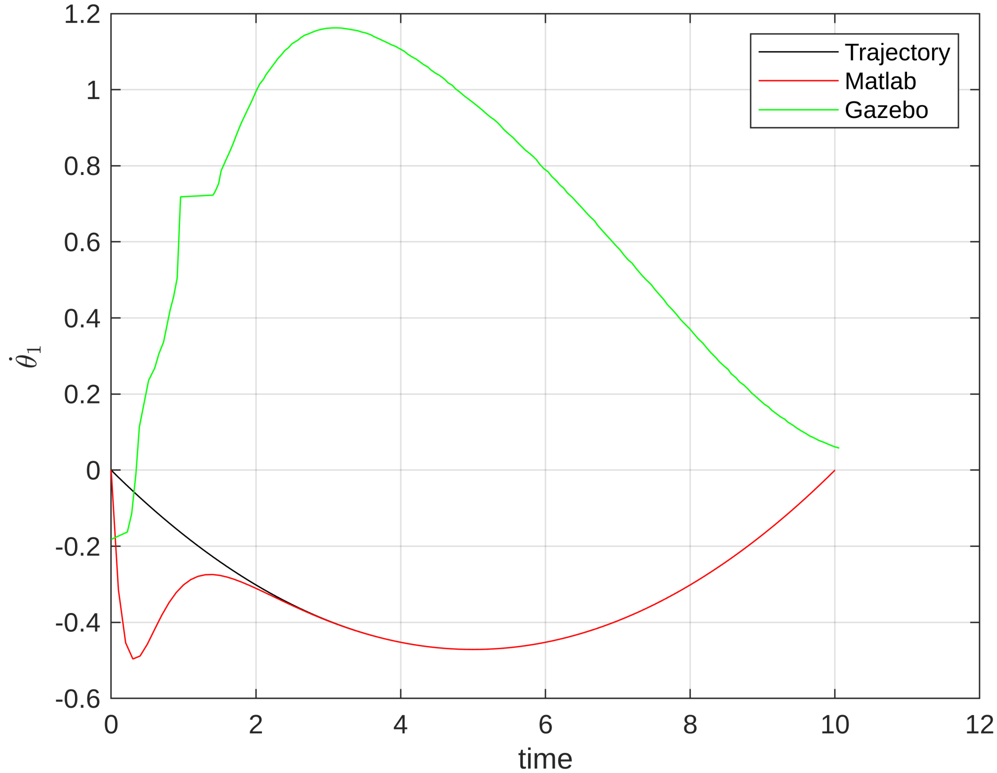
```
clc;
clear all;
close all;
matlab_efforts = readmatrix('matlab_efforts.xls');
matlab_joint_states = readmatrix('matlab_theta_plot_points.xls');
matlab_virtual_input = readmatrix('matlab_virtual_input.xls');
matlab_time = readmatrix('matlab_time.xls');
trajectory = readmatrix('trajectory.xls');
gazebo_efforts = readmatrix('gazebo_efforts.xls');
gazebo_joint_states = readmatrix('gazebo_theta_plot_points.xls');
gazebo_virtual_input = readmatrix('gazebo_virtual_input.xls');
gazebo_time = readmatrix('gazebo_time.xls');

plot(matlab_time, trajectory(:, 1), 'black')
hold on;
plot(matlab_time, matlab_joint_states(:, 1), 'red')
hold on;
plot(gazebo_time, gazebo_joint_states(:, 1), 'green')
legend('Trajectory','Matlab', 'Gazebo')
hold off;
grid on;
xlabel('time');
ylabel('\theta_{1}')
```

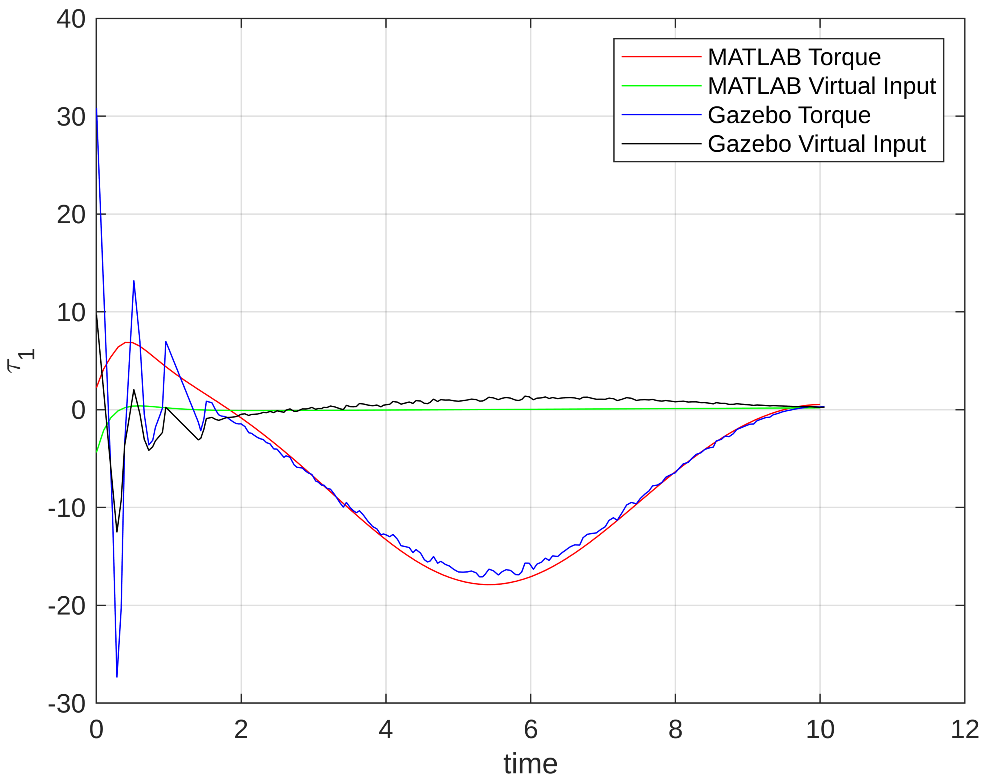
```
saveas(gcf, 'comparision_theta1.jpg')
```



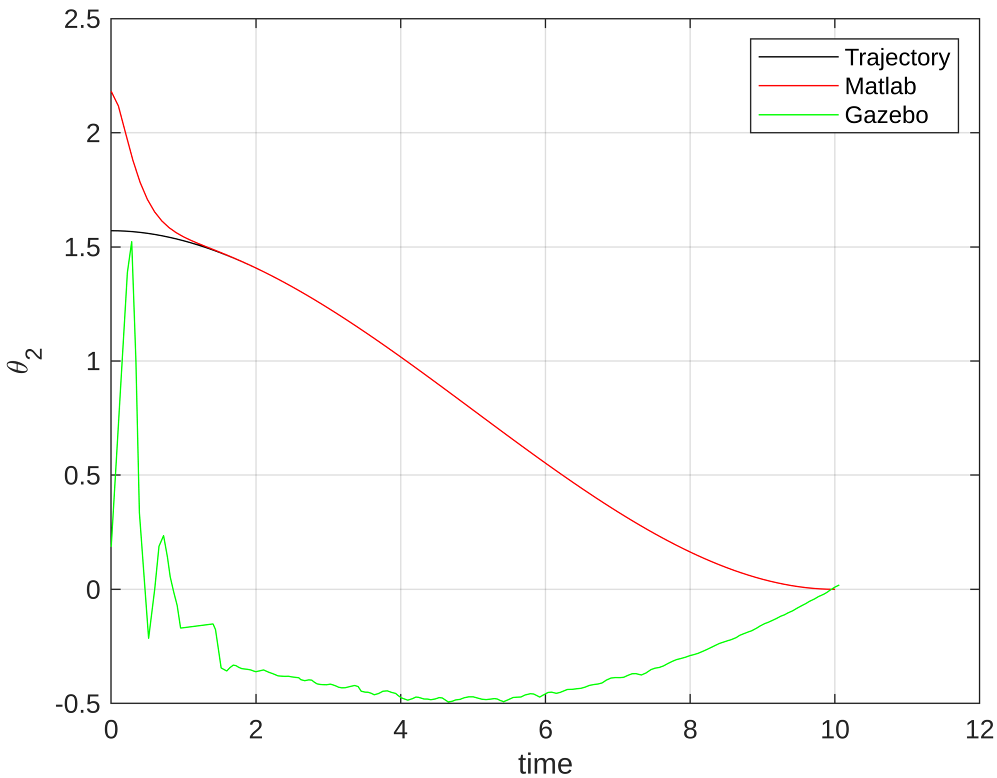

```
plot(matlab_time, trajectory(:, 2), 'black')
hold on;
plot(matlab_time, matlab_joint_states(:, 2), 'red')
hold on;
plot(gazebo_time, gazebo_joint_states(:, 2), 'green')
legend('Trajectory', 'Matlab', 'Gazebo')
hold off;
grid on;
xlabel('time');
ylabel('$\dot{\theta}_{1}$', 'Interpreter', 'latex')
saveas(gcf, 'comparision_dtheta1.jpg')
```



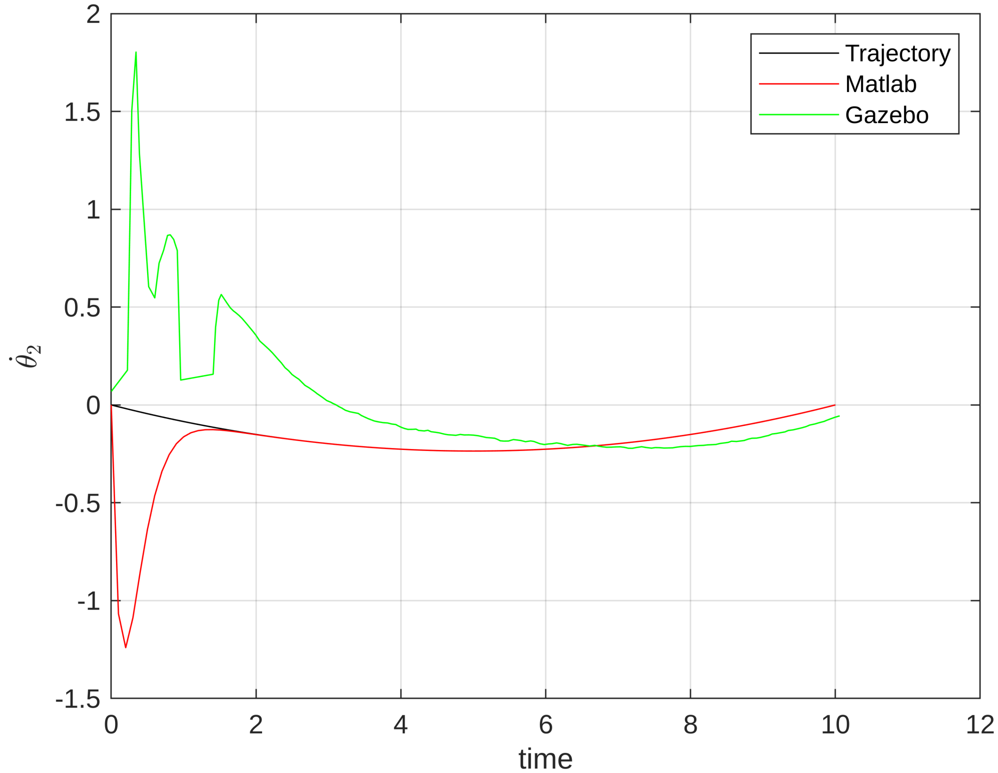

```
plot(matlab_time, matlab_efforts(:, 1), 'red')
hold on;
plot(matlab_time, matlab_virtual_input(:, 1), 'green')
plot(gazebo_time, gazebo_efforts(:, 1), 'blue')
plot(gazebo_time, gazebo_virtual_input(:, 1), 'black')
hold off;
legend('MATLAB Torque', 'MATLAB Virtual Input', 'Gazebo Torque', 'Gazebo Virtual Input'
grid on;
xlabel('time');
ylabel('\tau_{1}')
saveas(gcf, 'comparision_tau1.jpg')
```



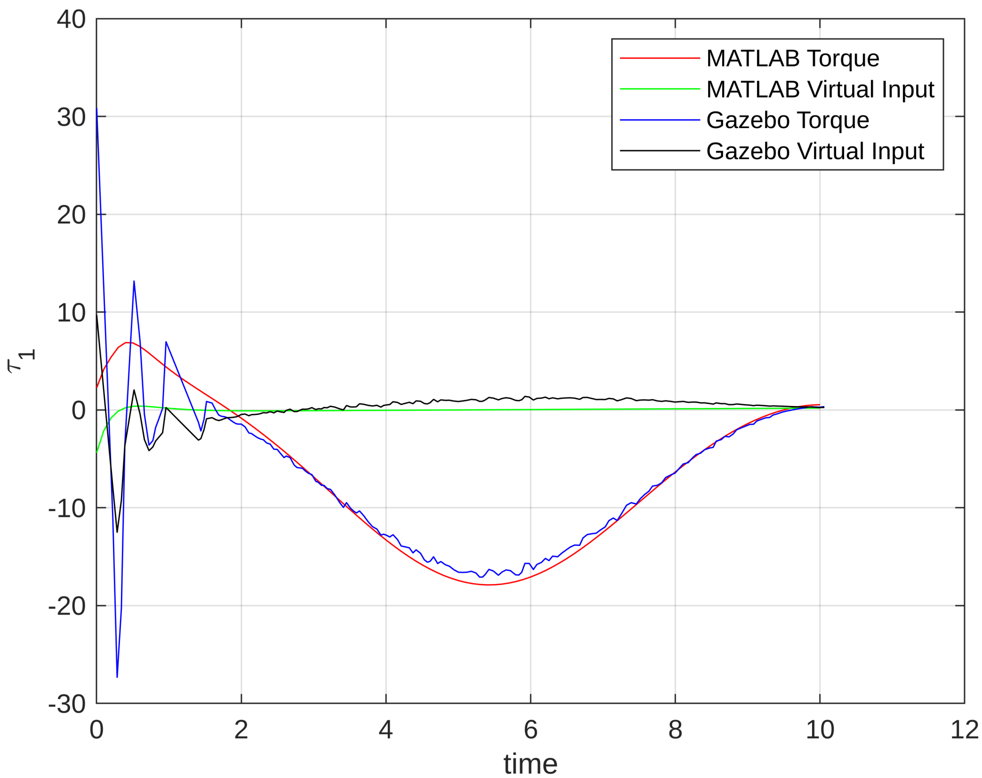

```
plot(matlab_time, trajectory(:, 3), 'black')
hold on;
plot(matlab_time, matlab_joint_states(:, 3), 'red')
hold on;
plot(gazebo_time, gazebo_joint_states(:, 3), 'green')
legend('Trajectory', 'Matlab', 'Gazebo')
hold off;
grid on;
xlabel('time');
ylabel('\theta_{2}')
saveas(gcf, 'comparision_theta2.jpg')
```




```
plot(matlab_time, trajectory(:, 4), 'black')
hold on;
plot(matlab_time, matlab_joint_states(:, 4), 'red')
hold on;
plot(gazebo_time, gazebo_joint_states(:, 4), 'green')
legend('Trajectory', 'Matlab', 'Gazebo')
hold off;
grid on;
xlabel('time');
ylabel('$\dot{\theta}_{2}$', 'Interpreter', 'latex')
saveas(gcf, 'comparision_dtheta2.jpg')
```




```
plot(matlab_time, matlab_efforts(:, 1), 'red')
hold on;
plot(matlab_time, matlab_virtual_input(:, 1), 'green')
plot(gazebo_time, gazebo_efforts(:, 1), 'blue')
plot(gazebo_time, gazebo_virtual_input(:, 1), 'black')
hold off;
legend('MATLAB Torque', 'MATLAB Virtual Input', 'Gazebo Torque', 'Gazebo Virtual Input'
grid on;
xlabel('time');
ylabel('\tau_{1}')
saveas(gcf, 'comparision_tau1.jpg')
```



40
30
20
10
0
-10
-20
-30
0
2
4
6
8
10
12
time
τ_1
MATLAB Torque
MATLAB Virtual Input
Gazebo Torque
Gazebo Virtual Input