MAE 3260
Fall 2025

Name: ARMageddon: Rise of the Robotic Limb

Topic of Interest: 2-link robotic arm

Abstract: The project involved the modelling of a 2-link robot arm. The arm has 2 servos, one attached to the base and the other connected at the joint, each driving each part of the arm individually. The input to the system was the angles to the horizontal for either arm. Aspects studied specifically include ODEs, transfer functions, block diagrams, and feedback control law.

List of MAE 3260 concepts or skills used in this group work:

- ODEs
- Transfer functions
- Block diagrams
- Feedback control law

By:
Alexis Barrow, Damayanti (Maya) Chakraborty, Mark Solis, Rachel Ortiz

Students/Roles:

Student	**Task/Role**	**Portfolio**
Name	**1-3 sentence summary of student's work and key take-aways**	**Link to results in student's individual portfolio**
Alexis Barrow	I generated the code and animation that represents the model, and assisted in the development of the ODEs. In addition to this, assisted in drawing the block diagrams, and worked on the overall formatting of the document. I learned a lot about tradeoffs with efficiency (especially considering controller selection), realism versus simplifications with modelling, and how to analyze a system given a set of assumptions.	https://cornell-mae-ug.github.io/fa25-portfolio-alexismbarrow/
Maya Chakraborty	I did the research on how to model the robotic arm, as well as the assumptions to simplify our model. I used the general model of robotic arms and applied our assumptions to them to derive our ODE. I learned the importance of	https://cornell-mae-ug.github.io/fa25-portfolio-maya-chakr/projects/MAE3260/

	choosing what controller you need depending on the application.	
Rachel Ortiz	I helped derive the ODEs, primarily focusing on the control law, and made the block diagrams. I learned about how complex actual robotic arms are.	https://cornell-mae-ug.github.io/fa25-portfolio-rmo59/projects/System%20Dynamics/
Mark Solis	I worked on deriving the model for the system using dynamics and any assumptions required to analyze the system. I helped with deriving the ODEs so that I can find the Laplace transforms and find the transfer functions of each link in the robot arm so that we can find the final positions of the robot arm given inputs from the servo in the pins of the arm.	

Date:
12/8/2025

Model Set-Up:

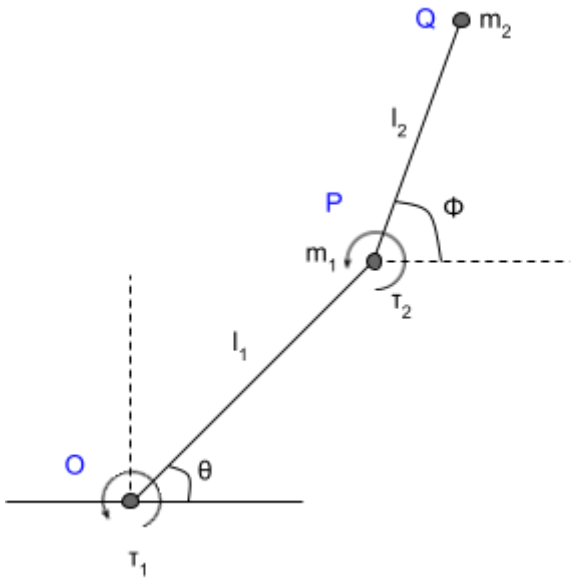

Figure 1: Simplified Diagram of Robotic Arm

Here, we assume a 2-dimensional, 2-linkage robotic arm. The linkages have lengths l_1 and l_2, and the nodes of the robot have masses m_1 and m_2. The robot will be given a desired θ_1 and Φ_2, and each node will move its respective linkage with a torque of T_1 and T_2.

Research:

The dynamic equations for a robotic arm can generally be represented by the following equation

$$\tau = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta)$$

where M is the mass/inertia matrix, C is the centrifugal matrix, and G is the gravity vector [1].

$$M(1,1) = (m_1 + m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 cos(\theta_2)$$

$$M(1,2) = m_2 l_2^2 + m_2 l_1 l_2 cos(\theta_2)$$

$$M(2,1) = m_2 l_2^2 + m_2 l_1 l_2 cos(\theta_2)$$

$$M(2,2) = m_2 l_2^2$$

$$C(1,2) = -2m_2 l_1 l_2 sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 - m_2 l_1 l_2 sin(\theta_2)\dot{\theta}_2^2$$

$$C(2,1) = m_2 l_1 l_2 sin(\theta_2)\dot{\theta}_1^2$$

$$G(1,1) = (m_1 + m_2)g l_1 sin(\theta_1) + m_2 g l_2 sin(\theta_1 + \theta_2)$$

$$G(2,1) = m_2 g l_2 sin(\theta_1 + \theta_2)$$

[1].

Assumptions and Simplification to Our Model:

The above model is too complicated for our purposes, so we simplified our model accordingly.

We assumed no gravity, which means $G(\theta) = 0$. Furthermore, we assumed that the robotic arm would only move one joint at a time. This means it would either move as a unit (both arms aligned, θ_1 changes but θ_2 remains constant. This means we can eliminate the Coriolis term and $C(\theta, \dot{\theta}) = 0$.

Lastly, we assume decoupled motion. M(1,1) and M(2,2) describe the inertia at joints 1 and 2, respectively. M(1,2) and M(2,1) represent the coupling of the system. Since we are describing decoupled motion, M(1,2) = M(2,1) = 0.

This leaves us with

$$\tau = M(\theta)\ddot{\theta}$$

$$\Rightarrow \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = [M(\theta)] \begin{bmatrix} \ddot{\theta} \\ \ddot{\Phi} \end{bmatrix}$$

where

$$M(1,1) = \frac{(m_1+m_2)(2l_1+l_2)^2}{3}$$

$$M(1,2) = 0$$

$$M(2,1) = \frac{m_2 l_2^2}{2}$$

$$M(2,2) = 0$$

This yields $\ddot{\theta} = \frac{3(\tau_0 - h_1\dot{\theta})}{3I + (m_1+m_2)(2l_1+l_2)^2}$ and $\ddot{\Phi} = \frac{3(\tau_p - h_2\dot{\Phi})}{2I + m_2 l_2^2}$

This can also be found using kinematics as derived in Figure 1 below.

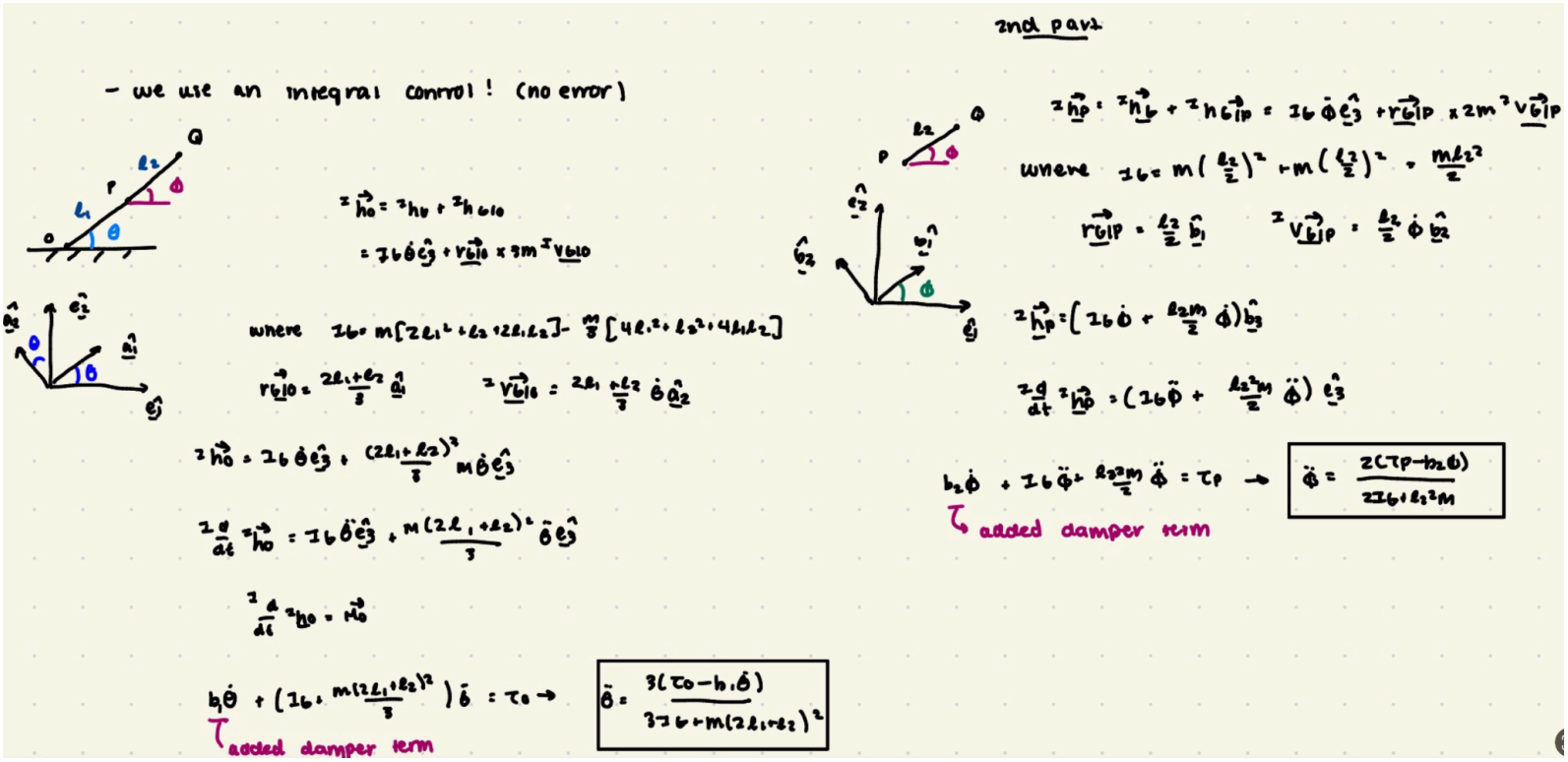

Figure 1: Derivation of ODEs using kinematics

This assumption can also be applied to the kinematics. Taking the moment of inertia as constant means that there would not be a dependence on the second arm's current position for the first arm's moment of inertia. This simplification is illustrated in Figure 2, with a strikethrough on the third term.

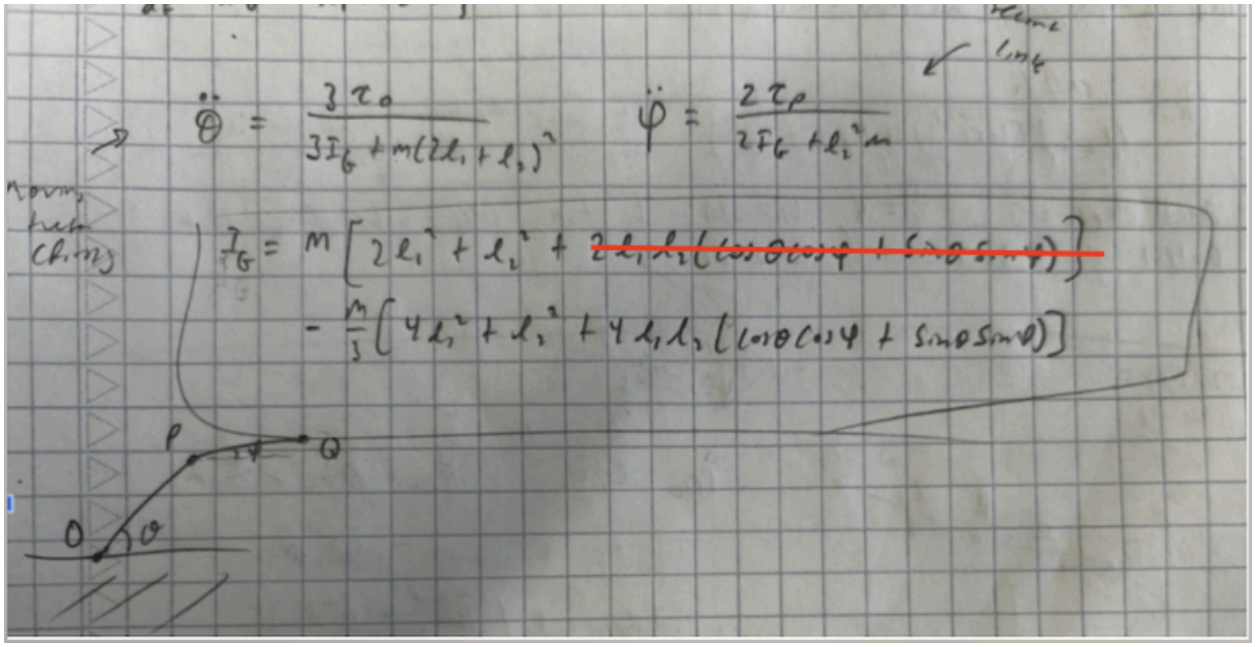

Figure 2: Derivation of Moment of Inertia Ig

Control Law

For simplicity, we looked to make as simple a controller as possible. We started by having each link have its own controller, ideally, both proportional controllers. However, in order to accommodate our earlier simplification of having both links be aligned for the initial movement, we needed the second link to be able to move to a position and have zero error. This resulted in the first link having a proportional (P) controller, while the second link has a proportional-integral (PI) controller.

Logistically, either controller was selected for functionality. While the P-controller Kp runs quicker, it yields some error in the end, so theta will be slightly incorrect. While the PI-controller yields no error, it runs much less quickly. To ensure both a timely and accurate robot, we implemented both. The reason the second arm was selected for having the PI controller was due to logistics on the use of the robot arm- should it be handing off an object, the user may be able to anticipate error from theta prior to the completion of phi, and adjust accordingly.

Block Diagram

Here are block diagrams for each link motor/controller:

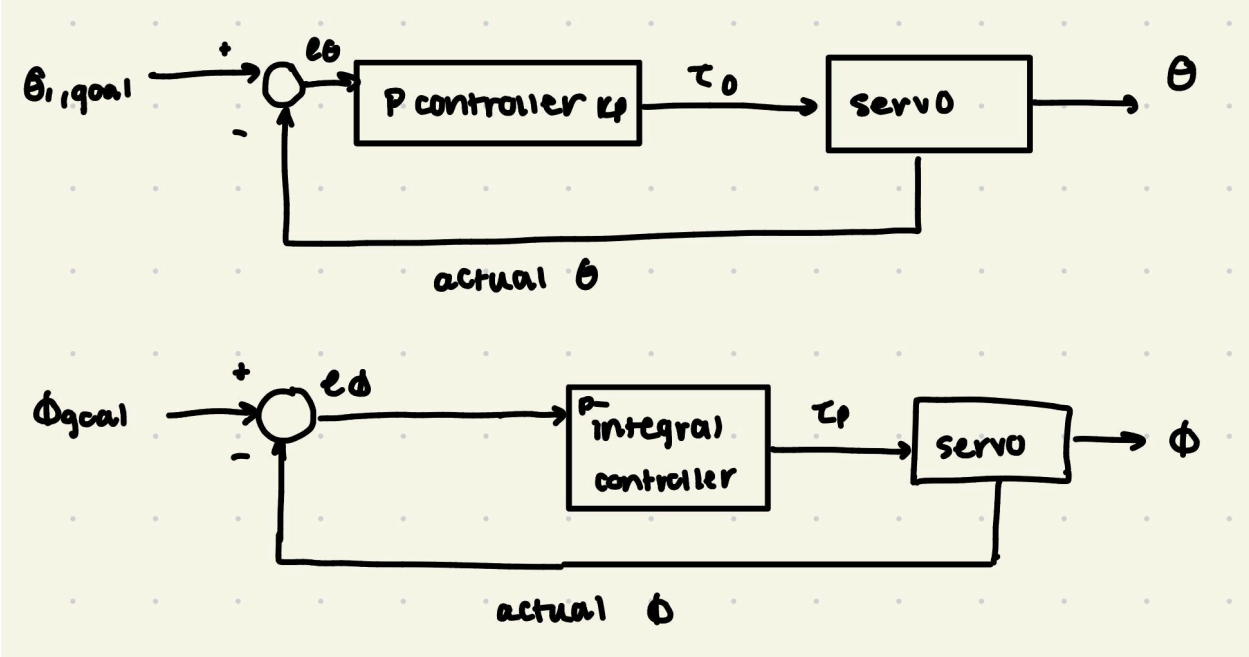

Figure 3: Block Diagram

Transfer Functions

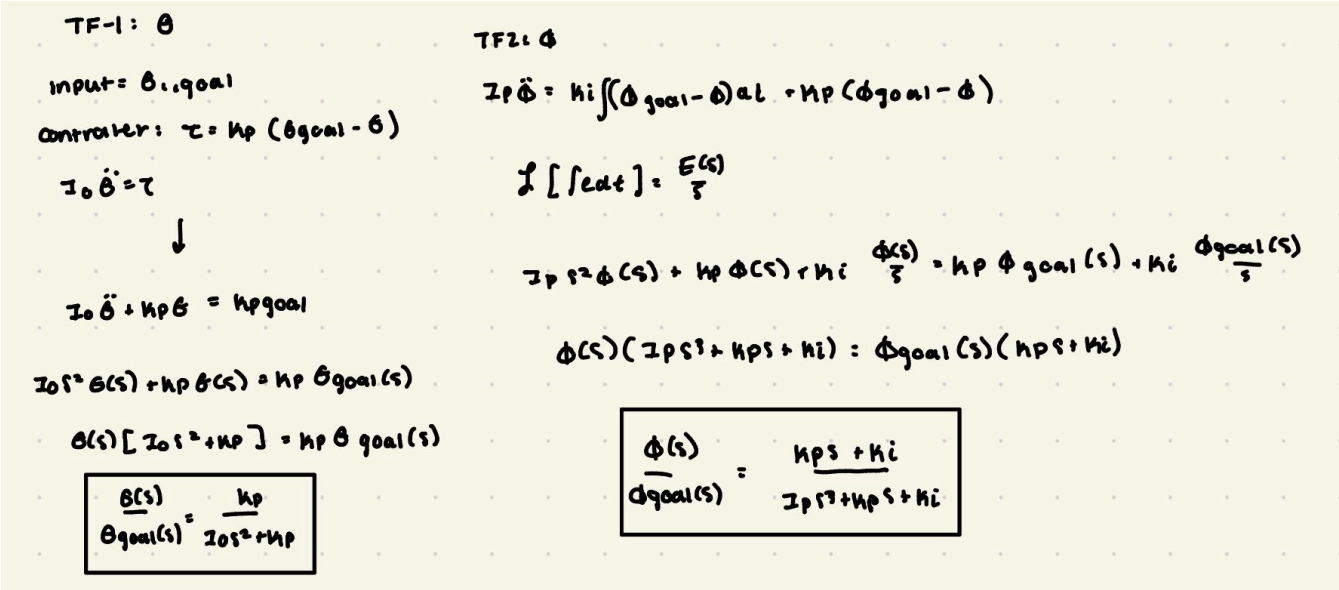

Figure 4: Derivation of transfer Functions

Modeling in MATLAB

A MATLAB script was generated with the assistance of ChatGPT to help model the system in motion. The script calls for 2 inputs, which are the angles of the first arm with respect to the horizontal and the second angle with respect to the horizontal. The output is an animation that displays the movement of the robot arm. Below is the code:

```
clear; clc; close all;

%% ------------------------------------------------------------
% Geometry
%% ------------------------------------------------------------
L1 = 4;    % length of first link
L2 = 2;    % length of second link

%% ------------------------------------------------------------
% Dynamics parameters
%% ------------------------------------------------------------
m1   = 1.0;    % mass of first link (kg)
m2   = 1.0;    % mass of second link (kg)

tau0 = 1.0;    % torque at base when first link moves (N·m)
taup = 1.0;    % torque at elbow when second link moves (N·m)

% (Optional) damping coefficients from your notes (not used in this
% constant-acceleration version, but here to show consistency)
b_theta = 0.0;
b_phi   = 0.0;

%% ------------------------------------------------------------
% Ask user for desired joint angles (deg)
%% ------------------------------------------------------------
```

```
theta_des_deg = input('Desired angle of FIRST link, theta (deg from horizontal): ');
phi_des_deg   = input('Desired angle of SECOND link, phi (deg, relative to first): ');

theta_des = deg2rad(theta_des_deg);
phi_des   = deg2rad(phi_des_deg);

%% ------------------------------------------------------------
% Inertia terms from the cleaned derivation
% Stage 1: both links move together about base O -> IO
%% ------------------------------------------------------------
% Link 1 about O (slender rod, pinned at one end)
I1O = (1/3)*m1*L1^2;

% Link 2 about its own center
I2G = (1/12)*m2*L2^2;

% Distance from O to COM of link 2
rG2 = L1 + L2/2;

% Link 2 about O (parallel-axis theorem)
I2O = I2G + m2*rG2^2;

% Total inertia about base for Stage 1
IO = I1O + I2O;

%% ------------------------------------------------------------
% Stage 2 inertia: link 2 about elbow P -> IP
%% ------------------------------------------------------------
IP = (1/3)*m2*L2^2;   % slender rod pinned at one end

%% ------------------------------------------------------------
% Constant angular acceleration magnitudes from τ = I * α
%% ------------------------------------------------------------
theta_ddot_mag = abs(tau0 / IO);   % |θ¨|
phi_ddot_mag   = abs(taup / IP);   % |φ¨|

% choose signs so motion goes toward the desired angles
theta_ddot = sign(theta_des) * theta_ddot_mag;
phi_ddot   = sign(phi_des)   * phi_ddot_mag;

%% ------------------------------------------------------------
% Stage 1: move ONLY the first link (second link rigid, torque at base)
%% ------------------------------------------------------------
% time to reach theta_des under constant acceleration
T1 = sqrt( 2*abs(theta_des) / theta_ddot_mag );
t1 = linspace(0, T1, 200);

theta1 = 0.5 * theta_ddot .* t1.^2;   % θ(t)
phi1   = zeros(size(t1));             % φ = 0 (no elbow motion yet)

%% ------------------------------------------------------------
% Stage 2: hold first joint fixed, move ONLY the second link
%% ------------------------------------------------------------
T2 = sqrt( 2*abs(phi_des) / phi_ddot_mag );
t2 = linspace(0, T2, 200);

theta2 = theta_des * ones(size(t2));  % base link fixed at final θ
```

```
phi2   = 0.5 * phi_ddot .* t2.^2;     % φ(t) relative to link 1

%% ---------------------------------------------------------
% Combine the two stages
%% ---------------------------------------------------------
t     = [t1, T1 + t2];
theta = [theta1, theta2];
phi   = [phi1,   phi2];

%% ---------------------------------------------------------
% Forward kinematics for each frame
%% ---------------------------------------------------------
x0 = 0; y0 = 0;

x1 = x0 + L1 .* cos(theta);
y1 = y0 + L1 .* sin(theta);

x2 = x1 + L2 .* cos(theta + phi);
y2 = y1 + L2 .* sin(theta + phi);

%% ---------------------------------------------------------
% Animation
%% ---------------------------------------------------------
figure;
axis equal;
grid on;
xlim([- (L1+L2)  (L1+L2)]);
ylim([- (L1+L2)  (L1+L2)]);
xlabel('x'); ylabel('y');

for k = 1:length(t)
    clf;
    hold on; grid on; axis equal;
    xlim([- (L1+L2)  (L1+L2)]);
    ylim([- (L1+L2)  (L1+L2)]);

    % draw links
    plot([x0 x1(k) x2(k)], [y0 y1(k) y2(k)], '-o', 'LineWidth', 2);

    title(sprintf('Two-link arm   t = %.2f s', t(k)));
    drawnow;
end
```

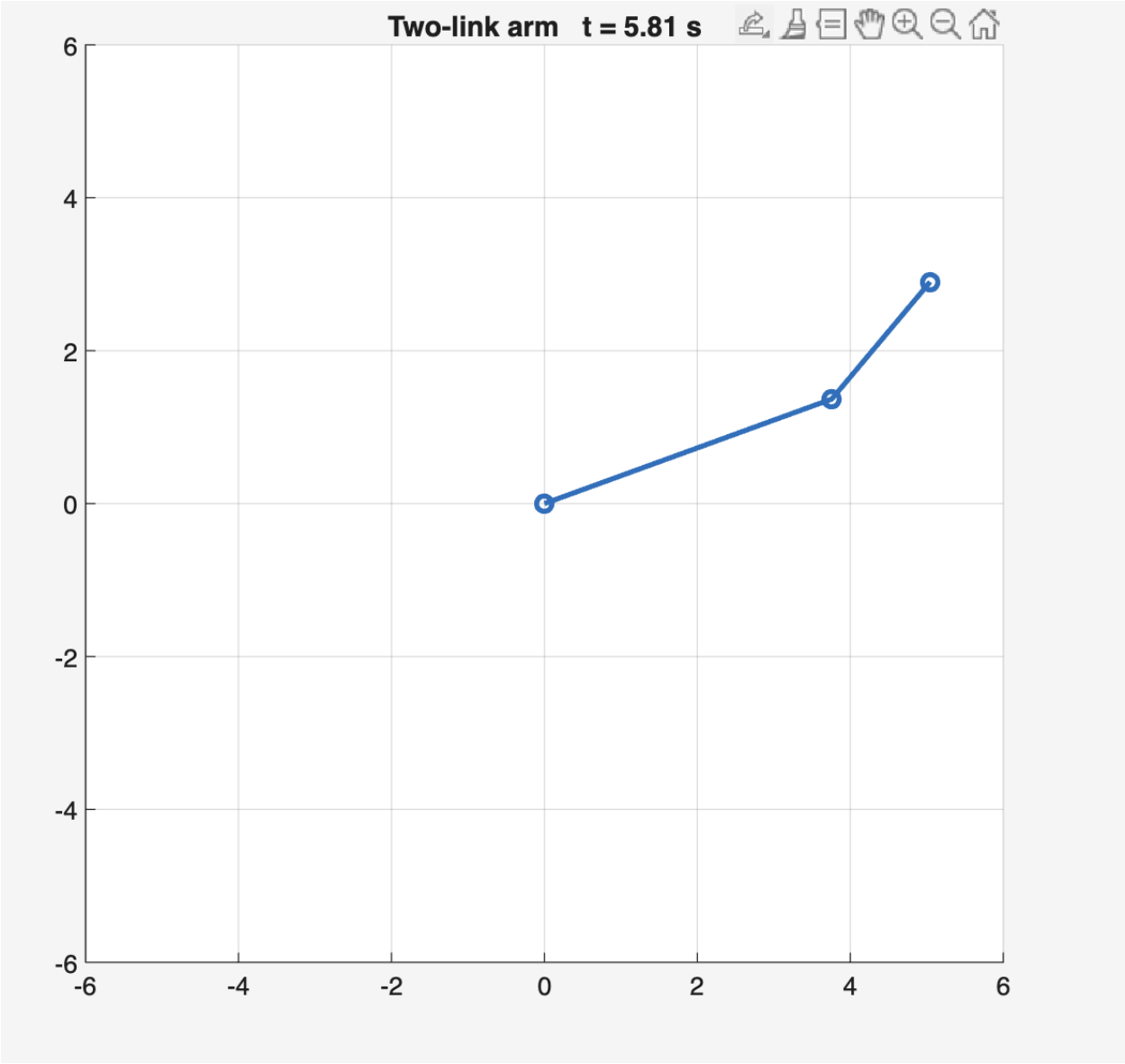

Figure 4: Example output with code for θ = 20 deg and ϕ= 30 deg

Reference List

[1] Okubanjo et al., "Modeling of 2-DOF Robot Arm and Control." *Futo Journal Series (FUTOJNLS)*, vol. 3, no. 2, Dec. 2017, pp. 80-92.