Modeling and Control of 2-DOF Robot Arm

Submitted in fullfillment of the requirements for the course

Introduction to Robotics

Submitted by:

Devin Yeung
Under the guidance of
Prof. Qingguo WANG

December 26, 2023

Abstract

This thesis delve into the control aspects of a two-degree-of-freedom robotic arm, focusing on PID control, kinematics, dynamics, and the design and implementation of controllers. The robot's parameters and key equations for forward and inverse kinematics are defined, followed by a detailed exploration of the dynamic model. The PID controller is introduced for torque control, with parameters tuned through simulations. Additionally, a Fuzzy Logic Controller is proposed as an alternative to PID, addressing challenges such as manual tuning complexity and system variability. The study provides valuable insights into the control of robotic arms, showcasing simulations and practical applications.

Keywords: *2-DOF, robotic arms, kinematics, dynamics, PID, fuzzy logic*

Contents

1. Robot Specification

We define a two degree of freedom robotic arm in Figure 1

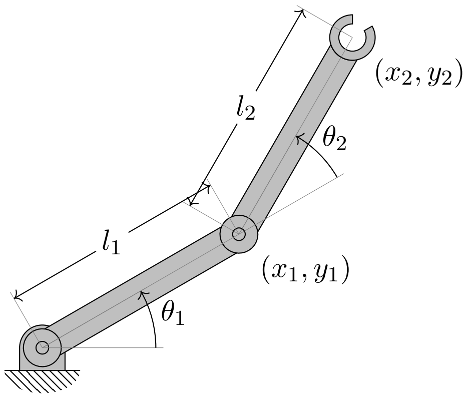

Figure 1: A 2-DOF robot arm

with the key parameters defined in Table 1

Parameter	**Symbol**
Length of the first link	l_1
Length of the second link	l_2
Mass of the first link	m_1
Mass of the second link	m_2
Rotation angle of the first link	θ_1
Rotation angle of the second link	θ_2

Table 1: Key parameters of the 2-DOF robot arm

2. Kinematics

Forward kinematics

The first thing to do is to define a suitable home configuration, in this case we set all joint angles to zero ($\theta_1 = \theta_2 = 0$), which is shown in Figure 2, then we perform following two rotations.

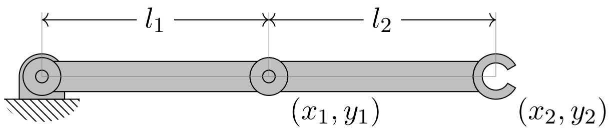

Figure 2: A 2-DOF robot arm at home position

First rotate joint 2 to θ_2 as shown in Figure 3.

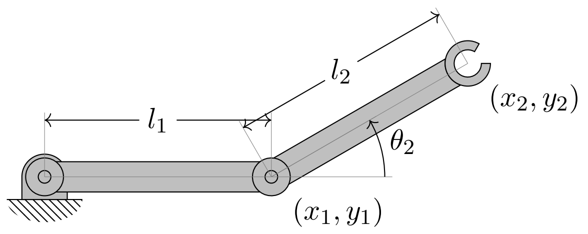

Figure 3: A 2-DOF robot arm at home configuration

Since the center of the rotation is (x_1, y_1), the rotation matrix is given in Equation (1)

$$A_2(\theta_2) = \begin{pmatrix} \cos(\theta_2) & -\sin(\theta_2) & (1-\cos\theta_2)l_1 \\ \sin(\theta_2) & \cos(\theta_2) & -l_1\sin\theta_2 \\ 0 & 0 & 1 \end{pmatrix} \tag{1}$$

Then we rotate the joint 1, which is similar to what we have done previously:

$$A_1(\theta_1) = \begin{pmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

By applying Equation (2) and Equation (1) respectively, the overall kinematic transformation matrix of the manipulator can be represented as:

$$\begin{aligned} K(\theta_1,\theta_2) &= A_1(\theta_1)A_2(\theta_2) \\ &= \begin{pmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & -l_1(\cos(\theta_1+\theta_2)-\cos(\theta_1)) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & -l_1(\sin(\theta_1+\theta_2)-\sin(\theta_1)) \\ 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{3}$$

To find the position of any point attached to the end-effector, we simply multiply its position vector in the home position by Equation (3):

$$\begin{aligned} \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} &= \begin{pmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & -l_1(\cos(\theta_1+\theta_2)-\cos(\theta_1)) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & -l_1(\sin(\theta_1+\theta_2)-\sin(\theta_1)) \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} l_1+l_2 \\ 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} l_2\cos(\theta_1+\theta_2)+l_1\cos(\theta_1) \\ l_2\sin(\theta_1+\theta_2)+l_1\sin(\theta_1) \\ 1 \end{pmatrix} \end{aligned} \tag{4}$$

The kinematic equations of the manipulator can be derived from the Equation (4):

$$\begin{aligned} x_2 &= l_2\cos(\theta_1+\theta_2)+l_1\cos(\theta_1) \\ y_2 &= l_2\sin(\theta_1+\theta_2)+l_1\sin(\theta_1) \end{aligned} \tag{5}$$

Inverse Kinematics

In inverse kinematics, the position of end effector (x_2, y_2) is known. We need to calculate the angle θ_1, θ_2 of each joint. First we determine the angle α, which is the angle between the end effector and the x-axis:

$$\alpha = \tan^{-1} \frac{y_2}{x_2} \tag{6}$$

Applying the law of cosines to the elbow angle β, which is the angle between link 1 and link 2, which yields

$$l_1^2 + l_2^2 - 2l_1l_2 \cos\beta = r^2 \tag{7}$$

where $r^2 = x_1^2 + y_1^2$, which yields:

$$\theta_2 = \pi - \beta = \pi - \cos^{-1} \frac{l_1^2 + l_2^2 - x_2^2 - y_2^2}{2l_1l_2} \tag{8}$$

Similarly

$$r^2 + l^2 - 2rl_1 \cos\gamma = l_2^2 \tag{9}$$

which yields

$$\theta_1 = \alpha - \gamma = \tan^{-1} \frac{y_2}{x_2} - \cos^{-1} \frac{x_2^2 + y_2^2 + l_1^2 - l_2^2}{2l_1\sqrt{x_2^2 + y_2^2}} \tag{10}$$

Since we have two possible configurations, elbow up and down, which is shown in Figure 4, we obtain:

$$\begin{aligned} \theta_1' &= \theta_1 + 2\gamma \\ \theta_2' &= -\theta_2 \end{aligned} \tag{11}$$

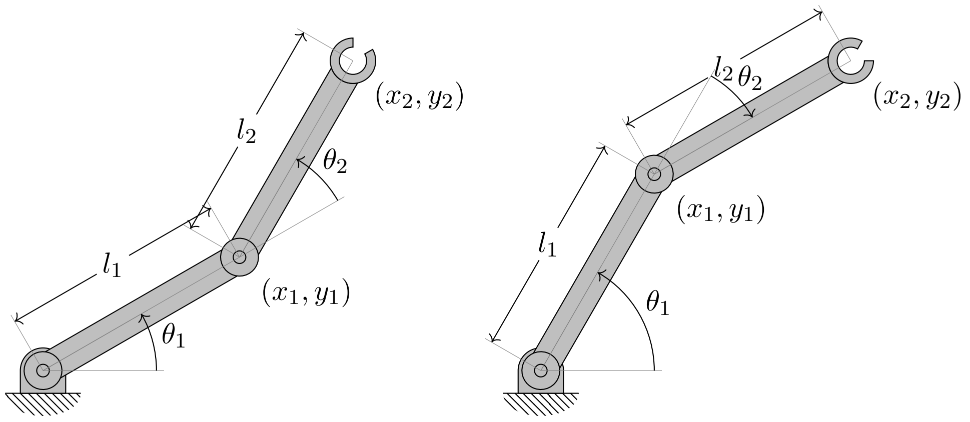

Figure 4: Two possible cases for 2-DOF robot arm inverse kinematics

3. Dynamics

Before we construct the dynamic model, following assumptions are made to simplify the model:

1. The actuators dynamics (motor and gear boxes) is not taken into account.
2. The effect of friction forces is assumed to be negligible

3. The mass of each link is assumed to be concentrated at the **end** of each link.

The dynamic model of a robot is concerned with the movement and the forces involved in the robot arm. We use Euler-Lagrangian method shown in Equation (12) to obtain the equation of motion.

$$F = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) - \frac{\partial \mathcal{L}}{\partial \theta} \tag{12}$$

where F is the external force, $\mathcal{L}$ is the Lagrangian equation of the system, which is given in Equation (13)

$$\mathcal{L}(\theta, \dot{\theta}) = K_E - P_E \tag{13}$$

To solve Lagrangian Equation (13), we need to calculate the kinematic energy K_E and the potential energy P_E respectively:

$$\begin{aligned} K_E &= \frac{1}{2}\left(m_1{v_1}^2 + m_2{v_2}^2\right) \\ P_E &= m_1 g y_1 + m_2 g y_2 \end{aligned} \tag{14}$$

The velocity of bob can be obtained by combining the transverse and longitudinal component velocities:

$$v_i^2 = {\dot{x}_i}^2 + {\dot{y}_i}^2 \quad i = 1, 2 \tag{15}$$

The position of mass is given by Equation (16), since we already assume that the weight of robotic arms is concentrated at the end of each robotic arm.

$$\begin{aligned} x_1 &= l_1 \cos(\theta_1) \\ y_1 &= l_1 \sin(\theta_1) \\ x_2 &= l_2 \cos(\theta_1 + \theta_2) + l_1 \cos(\theta_1) \\ y_2 &= l_2 \sin(\theta_1 + \theta_2) + l_1 \sin(\theta_1) \end{aligned} \tag{16}$$

By substituting Equation (16) and Equation (15) into Equation (14), we obtain:

$$\begin{aligned} K_E = &\frac{{\dot{\theta}_2}^2 {l_2}^2 m_2}{2} + \frac{{\dot{\theta}_1}^2 {l_1}^2 m_1}{2} + \frac{{\dot{\theta}_1}^2 {l_1}^2 m_2}{2} + \frac{{\dot{\theta}_1}^2 {l_2}^2 m_2}{2} \\ &+ \dot{\theta}_2 \dot{\theta}_1 {l_2}^2 m_2 + {\dot{\theta}_1}^2 l_1 l_2 m_2 \cos(\theta_2) + \dot{\theta}_2 \dot{\theta}_1 l_1 l_2 m_2 \cos(\theta_2) \end{aligned} \tag{17}$$

$$P_E = m_2 g \left(l_2 \sin(\theta_1 + \theta_2) + l_1 \sin \theta_1\right) + m_1 g l_1 \sin \theta_1 \tag{18}$$

Then we obtain

$$\begin{aligned} F_{\theta_1} = &\ddot{\theta}_2 {l_2}^2 m_2 + \ddot{\theta}_1 {l_1}^2 m_1 + \ddot{\theta}_1 {l_1}^2 m_2 + \ddot{\theta}_1 {l_2}^2 m_2 \\ &+ g l_2 m_2 \cos(\theta_1 + \theta_2) + g l_1 m_1 \cos(\theta_1) + g l_1 m_2 \cos(\theta_1) \\ &- {\dot{\theta}_2}^2 l_1 l_2 m_2 \sin(\theta_2) + \ddot{\theta}_2 l_1 l_2 m_2 \cos(\theta_2) + 2\ddot{\theta}_1 l_1 l_2 m_2 \cos(\theta_2) - 2\dot{\theta}_2 \dot{\theta}_1 l_1 l_2 m_2 \sin(\theta_2) \end{aligned} \tag{19}$$

we rewrite the Equation (19) in the form of:

$$\begin{aligned}F_{\theta_1} &= ((m_1+m_2)l_1^2 + m^2 l_2^2 + 2m_2 l_1 l_2 \cos\theta_2)\ddot{\theta}_1 \\ &+ (m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2)\ddot{\theta}_2 - m_2 l_1 l_2 \sin\theta_2 (2\dot{\theta}_1\dot{\theta}_2 + \dot{\theta}_2{}^2) \\ &+ (m_1+m_2) g l_1 \cos\theta_1 + m_2 g l_2 \cos(\theta_1+\theta_2)\end{aligned} \tag{20}$$

Similarly we can obtain F_{θ_2}

$$\begin{aligned}F_{\theta_2} &= \ddot{\theta}_2 {l_2}^2 m_2 + \ddot{\theta}_1 {l_2}^2 m_2 + g l_2 m_2 \cos(\theta_1+\theta_2) + \dot{\theta}_2{}^2 l_1 l_2 m_2 \sin(\theta_2) + \ddot{\theta}_1 l_1 l_2 m_2 \cos(\theta_2) \\ &= (m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2)\ddot{\theta}_1 + m_2 l_2^2 \ddot{\theta}_2 + m_2 l_1 l_2 \sin(\theta_2)\dot{\theta}_2{}^2 + m_2 g l_2 \cos(\theta_1+\theta_2)\end{aligned} \tag{21}$$

We can gather terms together into an equation of the form

$$F = M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) \tag{22}$$

with

$$M(\theta) = \begin{pmatrix} m_1 l_1^2 + m_2(l_1^2 + 2l_1 l_2 \cos\theta_2 + l_2^2) & m_2(l_1 l_2 \cos\theta_2 + l_2^2) \\ m_2(l_1 l_2 \cos\theta_2 + l_2^2) & m_2 l_2^2 \end{pmatrix} \tag{23}$$

$$c(\theta_1, \dot{\theta}) = \begin{pmatrix} -m_2 l_1 l_2 \sin\theta_2 (2\dot{\theta}_1\dot{\theta}_2 + \ddot{\theta}^2) \\ m_2 l_2 l_2 \dot{\theta}_1{}^2 \sin\theta_2 \end{pmatrix} \tag{24}$$

$$g(\theta) = \begin{pmatrix} (m_1+m_2) l_1 g \cos\theta_1 + m_2 g l_2 \cos(\theta_1+\theta_2) \\ m_2 g l_2 \cos(\theta_1+\theta_2) \end{pmatrix} \tag{25}$$

4. Controller Design

The input variable F in Equation (22) represents the torque applied to the robot, which is unknown. So it required a control in the force applied of the joints to reach the final state. For this particular case, we design two PID controls since the first arm motion is dependent from the second arm. (In fact, they still have a strong interactive) The PID law can written as:

$$F = K_P e + K_D \dot{e} + K_I \int e \, \mathrm{d}t \tag{26}$$

where K_P, K_I, K_D are proportional, integral and derivates gain of the PID controller, respectively, and e is the error term, given by:

$$e = \theta^d - \theta \tag{27}$$

$\theta^d = \begin{pmatrix} \theta_1^d \\ \theta_2^d \end{pmatrix}$ is the desired joint angle. The close-loop equation is obtained by substituting Equation (26) into Equation (22):

$$M(\theta)\ddot{\theta} + c(\theta, \dot{\theta}) + g(\theta) = K_P e + K_D \dot{e} + K_I \int e \, \mathrm{d}t \tag{28}$$

then we have

$$\ddot{\theta} = M(\theta)^{-1}(-c(\theta,\dot{\theta}) - g(\theta)) + \hat{F} \tag{29}$$

with

$$\hat{F} = M(\theta)^{-1}F \tag{30}$$

So, we decouple the system to have a new input

$$\hat{F} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \tag{31}$$

The error signals of the system are

$$\begin{aligned} e(\theta_1) &= \theta_{1f} - \theta_1 \\ e(\theta_2) &= \theta_{2f} - \theta_2 \end{aligned} \tag{32}$$

where θ_f is the final position. In our simulation, we set the initial position to:

$$\theta_0 = \begin{pmatrix} -\frac{\pi}{2} \\ \frac{\pi}{2} \end{pmatrix} \tag{33}$$

and the final position to:

$$\begin{pmatrix} \theta_{1f} \\ \theta_{2f} \end{pmatrix} = \begin{pmatrix} \frac{\pi}{2} \\ -\frac{\pi}{2} \end{pmatrix} \tag{34}$$

So the complete system equations can be written in the form of:

$$\ddot{\theta} = M(\theta)^{-1}(-c(\theta,\dot{\theta}) - g(\theta)) + \hat{F} \tag{35}$$

with

$$\hat{F} = \begin{pmatrix} K_{P_1}(\theta_{1f} - \theta_1) - K_{D_1}\dot{\theta}_1 + K_{i_1}\int e(\theta_1)\,\mathrm{d}t \\ K_{P_2}(\theta_{2f} - \theta_2) - K_{D_2}\dot{\theta}_2 + K_{i_2}\int e(\theta_2)\,\mathrm{d}t \end{pmatrix} \tag{36}$$

We do the simulation base on the previous defined model in MATLAB with the configuration in Table 2, the source code for replication can be found in Listing 2

Parameter	**Value**	**Parameter**	**Value**
m_1	1.0	m_2	1.0
l_1	1.0	l_2	1.0
initial θ_1	$-\frac{\pi}{2}$	initial θ_2	$\frac{\pi}{2}$
final θ_1	$\frac{\pi}{2}$	final θ_2	$-\frac{\pi}{2}$

Table 2: Configuration in the PID controller simulation

After tuning by trial and error we got with the PID controller parameters in Table 3:

Parameter	**Value**	**Parameter**	**Value**
K_{P_1}	40	K_{P_2}	20
K_{I_1}	7	K_{I_2}	7
K_{D_1}	10	K_{D_2}	10

Table 3: Key parameters of the PID controller

The simulation results are shown in Figure 5 and Figure 6 respectively.

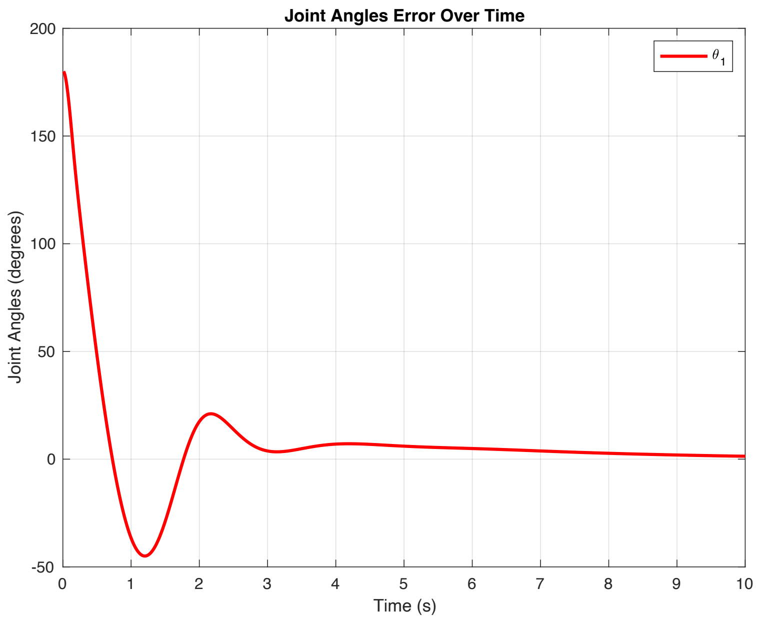

Figure 5: Joint 1 Angles Error Over Time

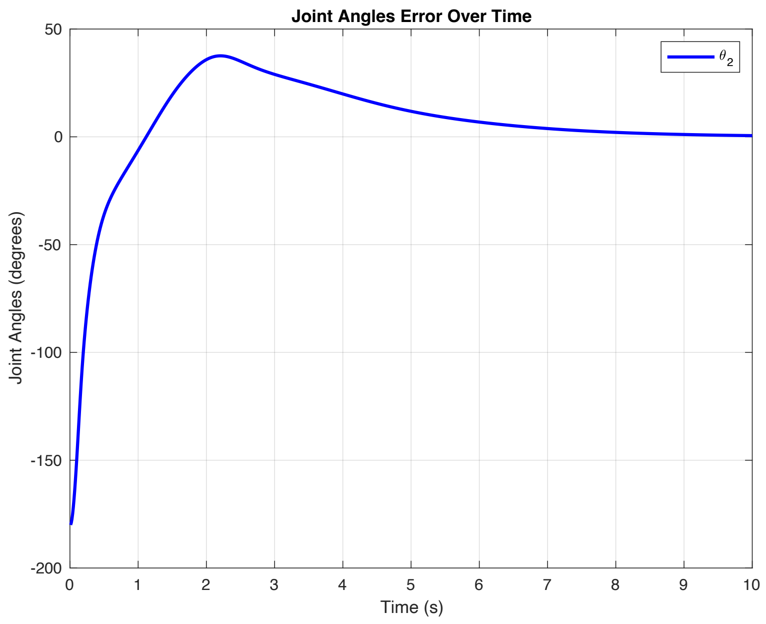

Figure 6: Joint 2 Angles Error Over Time

5. Fuzzy Logic Control

PID controllers, while widely used and effective in many applications, have some disadvantages related to tuning.

1. Manual Tuning Complexity: Tuning PID parameters manually can be a complex and time-consuming task.

2. System Variability: PID parameters that work well under one set of operating conditions may not be optimal for different conditions. System changes, such as variations in load or environmental conditions, can necessitate retuning.

Fuzzy control is a control technique that belongs to the expert systems and that allows for controlling dynamic systems without any mathematical model. This characteristic makes fuzzy control suitable for complex processes that are difficult to model analytically. The block diagram shown in Figure 7 represents the general structure of a fuzzy logic controller.

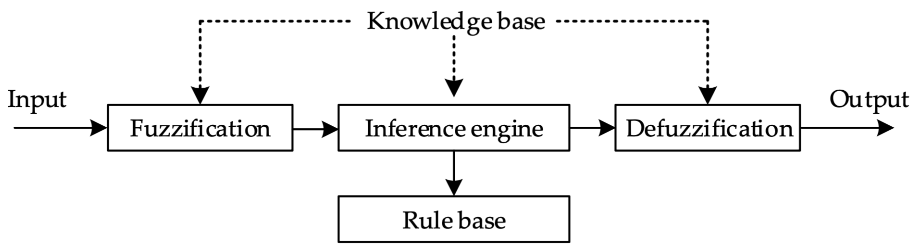

Figure 7: Fuzzy Logic Controller

This controller has three main components:

1. Fuzzification: Transforms input elements into membership degrees for linguistic terms in fuzzy sets, indicating the extent to which elements belong to each set.

2. Inference Engine: Makes decisions based on input data's membership degrees in fuzzy sets, using rules from the knowledge base. Outputs fuzzy sets calculated by the controller.

3. Defuzzification: Converts fuzzy values from the inference into useful values for the controlled process.

Fuzzy logic offers advantages in controlling a two-degree-of-freedom manipulator by providing adaptability to nonlinearities, robustness to varying conditions, and simplified tuning through a rule-based approach. Unlike PID controllers, fuzzy logic excels in handling complex and dynamic systems, allowing for effective control and improved performance under diverse operating conditions.

References

[1] N. M. Ghaleb and A. A. Aly. "Modeling and control of 2-DOF robot arm". In: *International Journal of Emerging Engineering Research and Technology* 6.11 (2018), pp. 24–31.

[2] J. M. Selig. *Introductory robotics*. Vol. 5. Prentice hall London, 1992.

[3] I. David and G. Robles. "PID control dynamics of a Robotic arm manipulator with two degrees of Freedom". In: *Control de Processos y Robotica* (2012), pp. 3–7.

[4] C. Urrea, J. Kern, and J. Alvarado. "Design and Evaluation of a New Fuzzy Control Algorithm Applied to a Manipulator Robot". In: *Applied Sciences* 10.21 (2020). ISSN: 2076-3417. URL: https://www.mdpi.com/2076-3417/10/21/7482.

[5] C. M. Lim and T. Hiyama. "Application of fuzzy logic control to a manipulator". In: *IEEE Transactions on Robotics and Automation* 7.5 (1991), pp. 688–691.

[6] K. Lochan and B. K. Roy. "Control of two-link 2-DOF robot manipulator using fuzzy logic techniques: a review". In: *Proceedings of Fourth International Conference on Soft Computing for Problem Solving: SocProS 2014, Volume 1*. Springer. 2014, pp. 499–511.

[7] K. M. Lynch and F. C. Park. *Modern robotics*. Cambridge University Press, 2017.

A. Matlab Simulation

A.1. Dynamics

Listing 1: Source code for dynamics verification

```
syms theta_1(t) theta_2(t) m_1 m_2 l_1 l_2 g

% kinematics
x_1 = l_1 * cos(theta_1(t))
y_1 = l_1 * sin(theta_1(t))
x_2 = l_2 * cos(theta_1(t) + theta_2(t)) + x_1
y_2 = l_2 * sin(theta_1(t) + theta_2(t)) + y_1

% v_1^2
v_1_sq = diff(x_1, t)^2 + diff(y_1, t)^2

% v_2^2
v_2_sq = diff(x_2, t)^2 + diff(y_2, t)^2

% kinematics energy
K_E = 1/2 * (m_1 * v_1_sq + m_2 * v_2_sq)

% potential energy
P_E = m_1 * g * y_1 + m_2 * g * y_2

% Lagrangian equation
L = K_E - P_E

% Force
F_1 = diff(diff(L,diff(theta_1(t),t)), t) - diff(L, theta_1(t))
F_2 = diff(diff(L,diff(theta_2(t),t)), t) - diff(L, theta_2(t))
```

A.2. PID control

Listing 2: Source code for PID simulation

```
function pid()

    % Define parameters
    l1 = 1; % length of the first link
    l2 = 1; % length of the second link
    m1 = 1; % mass of the first link
    m2 = 1; % mass of the second link
    g = 9.81; % acceleration due to gravity

```

```matlab
% Initial conditions
initial_theta = [- pi / 2, pi / 2];
desired_theta = [pi / 2, - pi / 2];

% PID gains for each link
Kp1 = 40;
Ki1 = 7;
Kd1 = 7;

Kp2 = 20;
Ki2 = 10;
Kd2 = 10;

% Simulation parameters
dt = 0.01; % time step
total_time = 10; % total simulation time

% Initialize variables
time = 0:dt:total_time;
num_steps = length(time);

theta = zeros(num_steps, 2);
theta(1, :) = initial_theta;
theta_err = zeros(num_steps, 2);
theta_dot = zeros(num_steps, 2);

% Initialize error variables for PID
error_integral1 = 0;
error_integral2 = 0;

for i = 1:num_steps
    % Calculate dynamics
    [M, C, G] = calculate_dynamics(theta(i, :), theta_dot(i, :), l1, l2, m1,
        m2, g);

    % Error calculation
    error1 = desired_theta(1) - theta(i, 1);
    error2 = desired_theta(2) - theta(i, 2);

    % PID control law
    u1 = Kp1 * error1 + Ki1 * error_integral1 + Kd1 * (0 - theta_dot(i, 1));
    u2 = Kp2 * error2 + Ki2 * error_integral2 + Kd2 * (0 - theta_dot(i, 2));

    % Update error integrals
    error_integral1 = error_integral1 + error1 * dt;
    error_integral2 = error_integral2 + error2 * dt;

    % Control input (torque)
    u = [u1; u2];

    % Solve for accelerations
```

```
        theta_ddot = M \ (u - C - G);

        % Update velocities and positions using Euler integration
        theta_dot(i+1, :) = theta_dot(i, :) + theta_ddot' * dt;
        theta(i+1, :) = theta(i, :) + theta_dot(i+1, :) * dt;
        theta_err(i+1, :) = [error1, error2];
    end

    % Plot results
    figure;
    plot(time(2:length(time)), rad2deg(theta_err(2:length(time), 1)), 'r', '
        LineWidth', 2);
    hold on;
    plot(time(2:length(time)), rad2deg(theta_err(2:length(time), 2)), 'b', '
        LineWidth', 2);
    title('Joint Angles Error Over Time');
    xlabel('Time (s)');
    ylabel('Joint Angles (degrees)');
    legend('\theta_1', '\theta_2');
    grid on;

end

function [M, C, G] = calculate_dynamics(theta, theta_dot, l1, l2, m1, m2, g)
    % Extract joint angles and velocities
    theta1 = theta(1);
    theta2 = theta(2);
    theta1_dot = theta_dot(1);
    theta2_dot = theta_dot(2);

    % Calculate necessary trigonometric terms
    c1 = cos(theta1);
    c2 = cos(theta2);
    s2 = sin(theta2);
    c12 = cos(theta1 + theta2);
    s12 = sin(theta1 + theta2);

    % Mass matrix M
    M = [m1 * l1^2 + m2 * (l1^2 + 2 * l1 * l2 * c2 + l2^2), m2 * (l1 * l2 * c2 +
         l2^2);
         m2 * (l1 * l2 * c2 + l2^2), m2 * l2^2];

    % Coriolis and centrifugal matrix C
    C = [-m2 * l1 * l2 * s2 * (2 * theta1_dot * theta2_dot + theta2_dot^2);
          m2 * l1 * l2 * theta1_dot^2 * s2];

    % Gravitational vector G
    G = [(m1 + m2) * l1 * g * c1 + m2 * g * l2 * c12;
         m2 * g * l2 * c12];
end
```