UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Modeling and Simulation of Mechatronic Systems
M

UR5 robotic arm model with Harmonic Drive actuation system

Professor:
Alessandro Macchelli

Student:
Luca Santoro

Academic year 2023/2024

Abstract

An industrial robot is a machine with significant characteristics of versatility and flexibility. In particular it is an electronically controlled mechanism that performs different tasks by interacting with external environment. These kind of robots are extensively used in the industrial manufacturing sector. The study of robotic manipulators mainly consists in defining the positions and orientations of the various links of which they are made up. In the industrial sector, robots play a crucial role in improving productivity and efficiency. Among these robots, robotic manipulators, especially those like the UR5, attract special attention. This study introduces a 3D model of the UR5 robotic manipulator, characterized by its 6 degrees of freedom. In particular, the manipulator is not treated as rigid but it has flexible joints, incorporating a harmonic drive system. Furthermore, a controller is designed to govern the robot manipulator equipped with the harmonic drive system. The entire system is implemented in Matlab/Simscape for validation and demonstration of results.

Contents

Introduction

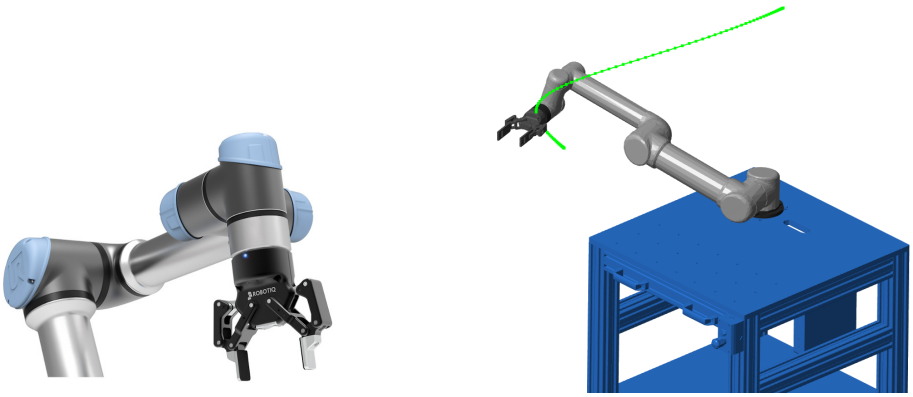
In this project, we created a model of an robotic arm Fig.1 with 6 degrees of freedom. To do this, we calculated homogeneous transformation matrices for each part of the arm, using default parameters known as 'DH' (Denavit-Hartenberg). We implemented the complete model using Matlab Simscape Multibody and Simulink tools. To control the robotic arm, we added a Proportional-Integrative-Derivative (PID) controller. Additionally, we have included modeling of devices called Harmonic Drive for each arm joint. To manage the dynamics of Harmonic Drive, we introduced an LQR controller. Finally, we generated a trapezoidal trajectory in the joint space to guide the arm movement. This project integrates complex concepts so we can control and simulate the robotic arm accurately and effectively.

Figure 1: Real Ur5 Robot Arm.

Chapter 1

UR5 robotic arm modelling

To move the 3D model in the space we use Simscape Multibody. Simscape Multibody provides a multibody simulation environment for 3D mechanical systems, such as robots. Whit this Matlab extension we import CAD geometries, masses, inertias, joints, constraints, and 3D geometries from PTC creo into our model. The final model is presented in Fig.1.1



(a) Real UR5 manipulator (b) UR5 in SimScape

Figure 1.1: UR5 manipulator.

In the following sections we will introduce the theory needed to build and control the UR5 model, while in the second chapter we will focus on the Harmonic Drives within each UR5 joint. In chapter three we will report all the results obtained from the simulations in MATLAB.

Motion control demands an accurate analysis of the characteristics of the mechanical structure, actuators, and sensors. The goal of such analysis is the derivation of the mathematical models describing the input/output relationship characterizing the robot components. Modelling a robot manipulator is therefore a necessary premise to finding motion control strategies [6].

1.1 Kinematic

Kinematic analysis of the mechanical structure of a robot concern the description of the motion with respect to a fixed reference Cartesian frame by ignoring the forces and moments that cause motion of the structure. It is meaningful to distinguish between *kinematics* and *differential kinematics*. With reference to a robot manipulator, kinematics describes the analytical relationship between the joint positions and the end-effector position and orientation. Differential kinematics describes the analytical relationship between the joint motion and the end-effector motion in terms of velocities, through the manipulator Jacobiann. The formulation of the kinematics relationship allows the study of two key problems of robotics, namely, the *direct kinematics problem* and the *inverse kinematics problem*. One of the most notable geometric features of the UR5 manipulator is its asymmetrical layout, which enhances the range of motion and optimizes the workspace. A key characteristic of this feature is that its last three joints do not intersect at a single point. Consequently, all six joints contribute to both the positioning and orientation of the end-effector, unlike other 6 DOF robotic manipulators where the first three joints are responsible for positioning and the last three for orientation. As a result, the kinematic analysis of UR5 is more complex compared to other manipulators.

1.1.1 Forward (Direct) Kinematics

Forward kinematics is the process of calculating the spatial position and orientation of a robotic manipulator's endeffector by incorporating relevant joint angles and link lengths associated with the manipulator's structure. To determine forward kinematics, we first need to assign coordinates to each joint and the end of the UR5 with the DH convention, and then connect each adjacent coordinate with transformation matrices. Briefly, *forward kinematics map joint coordinates to the end-effector pose*. To obtain forward kinematics of the robot manipulator, Homogenous Transformation matrices have to be calculated for each joint which includes translation and rotation matrices. In order to calculate Homogenous Transformation matrices, DH parameters of the robot arm have to be determined. In this case, these values are obtained from the datasheet of the robot manipulator. Consider an open-chain manipulator constituted by n + 1 links connected by n joints, where Link 0 is conventionally fixed to the ground. It is assumed that each joint provides the mechanical structure with a single DOF, corresponding to the joint variable. Since each joint connects two consecutive links, it is reasonable to consider first the description of kinematic relationship between consecutive links and then to obtain the overall description of manipulator kinematics in a recursive fashion. To this purpose, it is worth defining a coordinate frame attached to each link, from Link 0 to Link n. Then, the

coordinate transformation describing the position and orientation of Frame n with respect to Frame 0 (Fig. 1.2) is given by:

$$\mathbf{T}_n^0(q) = \mathbf{A}_1^0(q_1)\mathbf{A}_2^1(q_2)\dots\mathbf{A}_n^{n-1}(q_n) \tag{1.1}$$

The computation of the direct kinematics function is recursive and is obtained in a systematic manner by simple products of the homogeneous transformation matrices $A_i^{i-1}(q_i)$ for $i = 1, \dots, n$, each of which is a function of a single joint variable

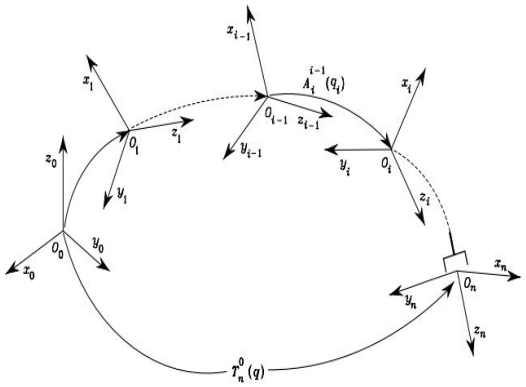


Figure 1.2: Coordinate transformations in an open kinematic chain.

In order to compute the direct kinematics equation for an open-chain manipulator according to the recursive expression in eq.1.1, a systematic, general method is to be derived to define the relative position and orientation of two consecutive links. It is convenient to set some rules also for the definition of the link frames [6]. With reference to Fig.1.3, let Axis i denote the axis of the joint connecting Link $i-1$ to Link i; the so-called Denavit-Hartenberg convention (DH) is adopted to define link Frame i:

- Choose axis z_i along the axis of Joint $i+1$.
- Locate the origin O_i at the intersection of axis z_i with the common normal to axes z_{i-1} and z_i. Also, locate $O_{i'}$ at the intersection of the common normal with axis z_{i-1}.
- Choose axis x_i along the common normal to axes z_{i-1} and z_i with direction from Joint i to Joint $i+1$.
- Choose axis y_i so as to complete a right-handed frame.

Once the link frames have been established, the position and orientation of Frame i with respect to Frame $i-1$ are completely specified by the following parameters:

- **Offset Distance** a_i : distance between z_i and z_{i-1} measured along x_i;
- **Translation distance** d_i: distance between axes x_i and x_{i-1} measured along the positive direction of z_{i-1};
- **Twist angle** α_i: between axes z_{i-1} and z_i. It is the angle required to rotate the axis z_{i-1} into alignment with the axis z_i in the right-hand sense about axis x_i;
- **Joint angle** θ_i : between axes x_{i-1} and x_i. It is the angle required to rotate the axis x_{i-1} into alignment with the axis x_i in the right-hand sense about axis z_{i-1}.

Two of the four parameters (a_i and α_i) are always constant and depend only on the geometry of connection between consecutive joints established by Link i.

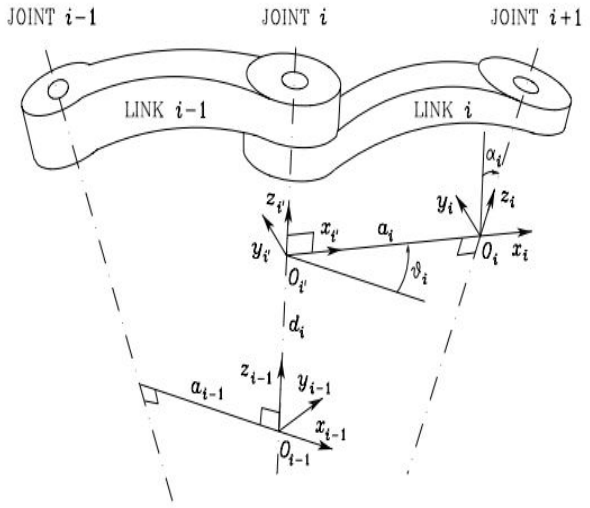


Figure 1.3: Standard Denavit-Hartenberg Kinematic parameters [6].

In Fig.1.4(a), the structure of UR5 manipulators and allocation of each joint is presented. It has six revolute joints and seven links, and each joint has a single degree of freedom. Based on the position of joints and the definition of DH convention, seven DH frames are assigned to the robot as Fig.1.4(b). Note that the position of joint frame $0_2x_2y_2z_2$ is not on the second link as required by modified DH convention, but on the first link, as this minimizes the number of non-zero DH parameters required and simplifies the subsequent derivation process.

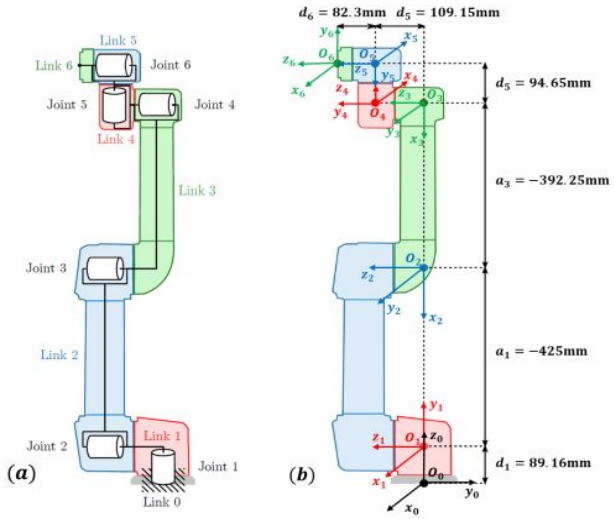


Figure 1.4: (a) Internal structure of the UR5 (b) DH frames assignment of UR5.

The link dimensions, provided by the manufacturer, are used to extract the DH parameters based on the assigned coordinate frames and link lengths. The resulting DH parameters for the specified frames can be found in Table.1.1.

DH Parameters				
Link i	$q_i[^\circ]$	d_i[mm]	a_i[mm]	$\alpha_i[^\circ]$
Base	q_1	0.089159	0	90
Shoulder	q_2	0	-0.425	0
Elbow	q_3	0	-0.39225	0
Wrist1	q_4	0.10915	0	90
Wrist2	q_5	0.09465	0	-90
Wrist3	q_6	0.0823	0	0

Table 1.1: DH table for UR5 manipulator.

The transformation matrix between every two adjacent joints can be expressed with Eq.1.2. Note that $c\theta$ and $s\theta$ represent $cos(\theta)$ and $sin(\theta)$.

$$A_i^{i-1} = \begin{bmatrix} R_i^{i-1} & p^{i-1} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} c(\theta_i) & -s(\theta_i)c(\alpha_i) & s(\theta_i)s(\alpha_i) & a_i c(\theta_i) \\ s(\theta_i) & c(\theta_i)c(\alpha_i) & -c(\theta_i)s(\alpha_i) & a_i s(\theta_i) \\ 0 & s(\alpha_i) & c(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

Substituting DH parameters in Table 1.1 into Equation 1.2, the transformation matrices between each coordinate frame can be obtained respectively:

$$A_1^0 = \begin{bmatrix} c\theta_1 & 0 & s\theta_1 & 0 \\ -s\theta_1 & 0 & -c\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{1.3}$$

$$A_2^1 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & a_2 c\theta_2 \\ s\theta_2 & c\theta_2 & 0 & a_2 s\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

$$A_3^2 = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & a_3 c\theta_3 \\ s\theta_3 & c\theta_3 & 0 & a_3 s\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.5}$$

$$A_4^3 = \begin{bmatrix} c\theta_4 & 0 & s\theta_4 & 0 \\ s\theta_4 & 0 & -c\theta_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.6}$$

$$A_5^4 = \begin{bmatrix} c\theta_5 & 0 & -s\theta_5 & 0 \\ s\theta_5 & 0 & c\theta_5 & 0 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.7}$$

$$A_6^5 = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ s\theta_6 & c\theta_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.8}$$

The overall transformation matrix for the forward kinematics of the UR5 manipulator is then obtained by multiplying the six transformation matrixes together:

$$T_6^0 = A_1^0 \cdot A_2^1 \cdot A_3^2 \cdot A_4^3 \cdot A_5^4 \cdot A_6^5 = \begin{bmatrix} R_6^0 & P^0 \\ 0 & 1 \end{bmatrix} \tag{1.9}$$

Rotational matrix, R_6^0 and position vector, P^0 indicate the orientation and position of the end-effector in the base coordinates, respectively. As there is no dedicated tool installed at the end of the last link, the original point O_6 is regarded as the tip of the end-effector. The position vector of 0_6 in terms of joint frame $O_6x_6y_6z_6$ can be represented by $P^6 = [0, 0, 0, 1]^T$. Therefore, the position of this point in the base frame could be calculated as:

$$P^0 = T_6^0 P^6 = \begin{bmatrix} n_x & o_x & a_x & p_y \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} d_5c_1d_{234} + d_4s_1 - d_6c_1c_{234} + a_2c_1d_2 + d_6c_5s_1 + a_3c_1c_2c_3 - a_3c_1s_2s_3 \\ d_5s_1sin_{234} - d_4c_1 - d_6s_1c_{234} - d_6c_1c_5 + a_2c_2s_1 + a_3c_2c_3s_1 - a_3s_1s_2s_3 \\ d_1 - d_6s_{234}s_5 + a_3s_{23} + a_2s_2 - d_5c_{234} \\ 1 \end{bmatrix} \tag{1.10}$$

Where c_{234} represents the $cos(\theta_2 + \theta_3 + \theta_4)$ and s_{234} stands for the $sin(\theta_2 + \theta_3 + \theta_4)$.

1.2 Trajectory Planning

With reference to the tasks assigned to a manipulator, the issue is whether to specify the motion at the joints or directly at the end-effector. *In material handling tasks, it is sufficient to assign only the pick-up and release locations of an object (point-to-point motion), whereas, in machining tasks, the end-effector has to follow a desired trajectory (path motion).* The goal of trajectory planning is to generate the timing laws for the relevant variables (joint or end-effector) starting from a concise description of the desired motion [6].

1.2.1 Joint Space and Operational Space

As described previously, the direct kinematics equation of a manipulator allows the position and orientation of the end-effector frame to be expressed as a function of the joint variables with respect to the base frame. If a task is to be specified for the end-effector, it is necessary to assign the end-effector position and orientation, eventually as a function of time (trajectory). The position can be given by a minimal number of coordinates with regard to the geometry of the structure, and the orientation can be specified in terms of a minimal representation (Euler angles) describing the rotation of the end-effector frame with respect to the base frame [6]. In this way, it is possible to describe the end-effector pose by means of the $(m \times 1)$ vector, with $m \leq n$.

$$\mathbf{x}_e = \begin{bmatrix} \mathbf{p}_e \\ \Phi \end{bmatrix} \tag{1.11}$$

The $\mathbf{p}_e$ describes the end-effector position and Φ its orientation. The vector $\mathbf{x}_e$ is defined in the space in which the manipulator task is specified; hence, this space is typically called *operational space*. On the other hand, the *joint space* (configuration space) denotes the space in which the $(n \times 1)$ vector of joint variables:

$$\mathbf{q}_e = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix} \tag{1.12}$$

is defined; it is $q_i = \theta_i$ for a revolute joint and $q_i = d_i$ for a prismatic joint. the direct kinematics equation can be written in a form other than 1.1, i.e.,

$$\mathbf{x}_e = K(q) \tag{1.13}$$

The $(m \times 1)$ vector function $K()$ - nonlinear in general - allows computation of the operational space variables from the knowledge of the joint space variables.

Workspace

With reference to the operational space, an index of robot performance is the so-called workspace; this is the region described by the origin of the end-effector frame when all the manipulator joints execute all possible motions. It is often customary to distinguish between *reachable workspace* and **dexterous workspace**. The latter is the region that the origin of the end-effector frame can describe while attaining different orientations, while the former is the region that the origin of the end-effector frame can reach with at least one orientation. Obviously, the dexterous workspace is a subspace of the reachable workspace. A manipulator with less than six DOFs cannot take any arbitrary position and orientation in space. For an n-DOF manipulator, the reachable workspace is the geometric locus of the points that can be achieved by considering the direct kinematics equation for the sole position part, i.e,

$$\mathbf{p}_e = \mathbf{p}(\mathbf{q}) \quad q_{im} \leq q_i \leq q_{iM} \quad for \ i = 1, \ldots, n, \tag{1.14}$$

where $q_{im}(q_{iM})$ denotes the minimum (maximum) limit at Joint i. This volume is finite, closed, connected - $\mathbf{p}(\mathbf{q})$ is a continuous function - and thus is defined by its bordering surface.

1.2.2 Desired Trajectory

In the project to define the desired trajectory, which the robot must reach, the Peter Corke Robotics Toolbox is used [1]. Starting from a Joint Space, the initial configuration chosen is always the same. In the code we indicate it as $\mathbf{q}_z$, it corresponds to the configuration of the zero angle of the joints Fig.1.5, while $\mathbf{q}_n$ represents the desired final position.

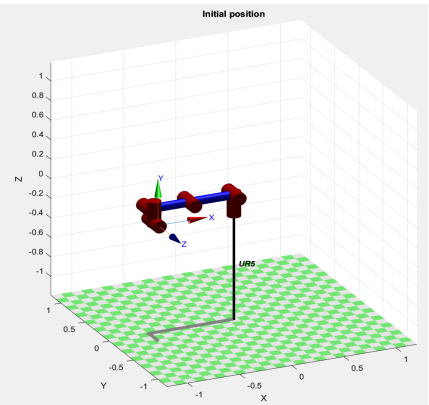


Figure 1.5: Initial configuration, zero joint angle configuration.

To calculate the *joint space trajectory* from an initial position (q_z) to the desired position (q_n), the function *jtraj()* is used. The joint angles were chosen taking the figure 1.6 as reference

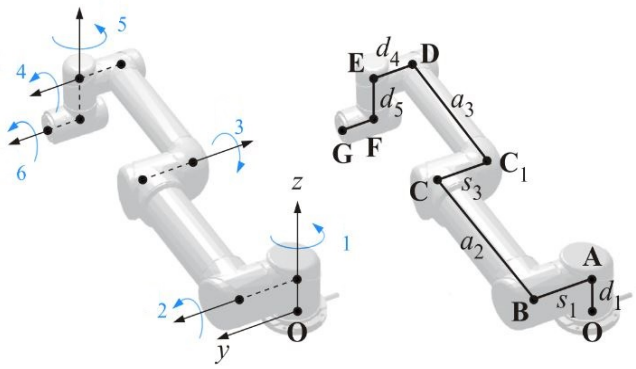
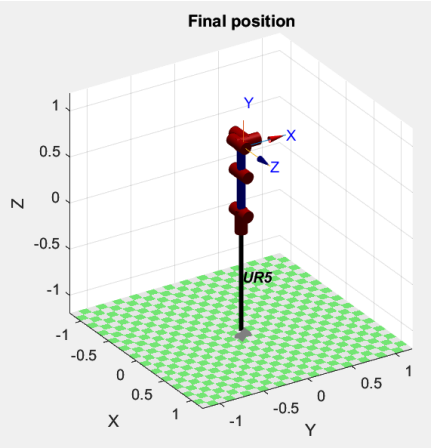
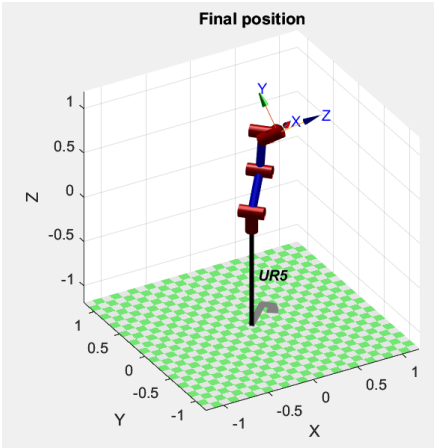


Figure 1.6: Joint angles.

Different trajectories with different desired positions were tested, only two of them are considered below:



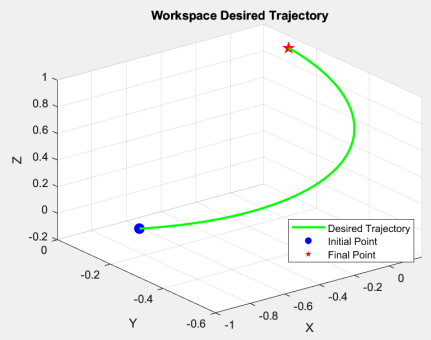
(a) Desired final configuration 1 $\mathbf{q}_n = \begin{bmatrix} \pi/2 & -\pi/2 & 0 & 0 & 0 & \pi/2 \end{bmatrix}$



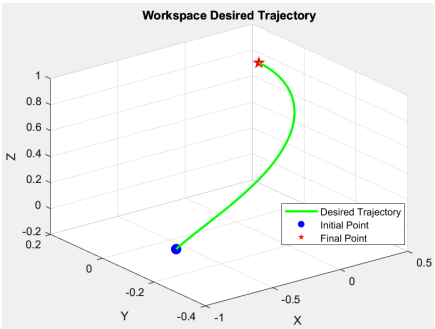
(b) Desired final configuration 2 $\mathbf{q}_n = \begin{bmatrix} \pi/4 & -3\pi/4 & \pi/6 & 0 & \pi/6 & \pi/2 \end{bmatrix}$

Figure 1.7: Two different desired final configurations.

Considering the previous desired final configurations (joint space), the corresponding two *Workspace Trajectories* are obtained, using the Forward kinematic function *fkine()*:



(a) Desired Trajectory 1



(b) Desired Trajectory 2

Figure 1.8: Desired Workspace trajectories.

The previous ones are the trajectories that the End Effector of the manipulator must follow.

1.3 Inverse Kinematics Problem

The direct kinematics equation, either in the form 1.1 or in the form 1.13, establishes the functional relationship between the joint variables and the end-effector position and orientation. *The inverse kinematics problem consists of the determination of the joint variables corresponding to a given*

end-effector position and orientation. The solution to this problem is of fundamental importance in order to transform the motion specifications, assigned to the end-effector in the operational space, into the corresponding joint space motions that allow execution of the desired motion. The inverse kinematics problem is much more complex for the following reasons:

- The equations to solve are in general nonlinear, and thus it is not always possible to find a closed-form solution.
- Multiple solutions may exist.
- Infinite solutions may exist, e.g., in the case of a kinematically redundant manipulator.
- There might be no admissible solutions, in view of the manipulator kinematic structure.

For a six-DOF manipulator without mechanical joint limits, there are in general up to 16 admissible solutions.

1.4 Dynamics

Derivation of the dynamic model of a manipulator plays an important role for simulation of motion, analysis of manipulator structures, and design of control algorithms. There are two possible methods to obtain the equations of motion of a robot arm in the joint space. The first method is based on the *Lagrange formulation* and is conceptually simple and systematic. The second method is based on the *Newton-Euler formulation* and yields the model in a recursive form; it is computationally more efficient since it exploits the typically open structure of the manipulator kinematic chain [6]. Manipulator dynamics is usually described by the well-known expression 1.15 for a generic n-DOF manipulator

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \tau \tag{1.15}$$

where $\mathbf{B(q)} \in \mathbb{R}^{n \times n}$ is the inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n \times n}$ is the Coriolis matrix and $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^{n \times 1}$ is the gravitational force vector; $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}} \in \mathbf{R}^{n \times n}$ are respectively joints position, velocity and acceleration. The right hand side of 1.15 is the input torque/force vector $\tau \in \mathbf{R}^{n \times 1}$. The equation 1.15 describes the manipulator rigid-body dynamics and is known as the inverse dynamics - given the pose, velocity and acceleration it computes the required joint forces or torques. These equations can be derived using any classical dynamics method such as Newton's second law and Euler's equation of motion or a Lagrangian energy-based approach.

1.4.1 Mass Distribution

Consider the links that compose the UR5 manipulator as rigid bodies. During the operation of the UR5, these rigid bodies perform rotational motion around varying axes within a three dimensional space. In such situations, the inertia tensor is frequently utilized to represent the mass distribution of the links. The inertia tensor can be expressed as:

$$I_i = \begin{bmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{xy} & I_{yy} & -I_{yz} \\ -I_{xz} & -I_{yz} & I_{zz} \end{bmatrix} \tag{1.16}$$

Where I_i represents the inertia tensor of link i in the coordinate frame i. The elements I_{xx}, I_{yy} and I_{zz} are called the mass moments of inertia and the other elements with mixed indices are called the mass products of inertia. The elements of inertia tensor can be calculated with the equations as follow:

$$\begin{aligned} I_{xx} &= \iiint (y^2 + z^2)\rho(x,y,z)dxdydz \quad I_{xy} = \iiint xy\rho(x,y,z)dxdydz \\ I_{yy} &= \iiint (x^2 + z^2)\rho(x,y,z)dxdydz \quad I_{xz} = \iiint xz\rho(x,y,z)dxdydz \\ I_{zz} &= \iiint (x^2 + y^2)\rho(x,y,z)dxdydz \quad I_{yz} = \iiint yz\rho(x,y,z)dxdydz \end{aligned} \tag{1.17}$$

Therefore, the mass distribution characteristics of each link in the robotic system can be fully determined by knowing the position of the center of mass and the inertia tensor. The mass properties of the UR5, are presented in Table 1.2, which includes the total mass of each link and their corresponding center point vector. Additionally, the official positions of the mass centers are illustrated in Figure 1.9.

Parameters		
Link	m_i[Kg]	$r_{G,i}$[m]
1	3.7	$(0.0, -0.02561, 0.00193)^T$
2	8.393	$(0.2125, 0.0, 0.1136)^T$
3	2.33	$(0.15, 0.0, 0.0265)^T$
4	1.1219	$(0.0, -0.0018, 0.01634)^T$
5	1.1219	$(0.0, 0.0018, 0.01634)^T$
6	0.1879	$(0.0, 0.0, -0.001159)^T$

Table 1.2: Mass and position of the center of gravity of each link.

As seen in Figure 1.9a, the links of UR5 exhibit irregular and asymmetrical structures, with their mass being non-uniformly distributed. This makes

it challenging to directly compute their inertia tensors using Equation 1.17. Therefore, the idea is to use a simplified model to address this complexity.

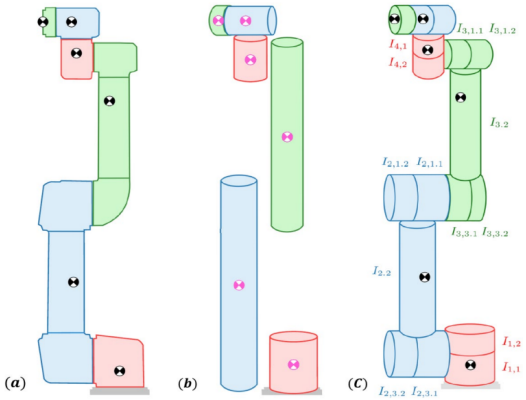


Figure 1.9: (a) Official mass center points from the manufacturer. (b) Mass center points from the cylindrical model. (c) Mass center points from Kufieta's model.

A common model is to approximate the links as cylinders with homogeneous density, as shown in Figure 1.9b. However, this model's centroids do not align with the mass centers provided by the manufacturer for UR5, resulting in inaccurate dynamic equations. The primary reason is that the UR5 is a manipulator with lightweight design, and the location of the motors significantly influences the mass distribution of the links, thus the links can not be simply regarded as cylinders. To overcome this issue, Kufieta [3] developed an improved model by dividing each link into multiple cylinders with varying homogeneous densities and highlighting the contribution of motors to mass. This approach, shown in Figure 1.9c, successfully preserves the correct link shape while ensuring the location of the mass center point aligns with the manufacturer's data under the assumption of homogeneous density for individual cylinders. With this refined model, the *inertia tensors* of the six links can be obtained using Equation 1.17, as well as the mass properties presented in Table 1.2. The results are as follows:

$$
\begin{aligned}
I_1 &= \begin{bmatrix} 0.0067 & 0 & 0 \\ 0 & 0.0064 & 0 \\ 0 & 0 & 0.0067 \end{bmatrix} & I_2 &= \begin{bmatrix} 0.0149 & 0 & 0 \\ 0 & 0.3564 & 0 \\ 0 & 0 & 0.3553 \end{bmatrix} \\
I_3 &= \begin{bmatrix} 0.0025 & 0 & 0.0034 \\ 0 & 0.0551 & 0 \\ 0.0034 & 0 & 0.0546 \end{bmatrix} & I_4 &= \begin{bmatrix} 0.0012 & 0 & 0 \\ 0 & 0.0012 & 0 \\ 0 & 0 & 0.0009 \end{bmatrix} \\
I_5 &= \begin{bmatrix} 0.0012 & 0 & 0 \\ 0 & 0.0012 & 0 \\ 0 & 0 & 0.0009 \end{bmatrix} & I_6 &= \begin{bmatrix} 0.0001 & 0 & 0 \\ 0 & 0.0001 & 0 \\ 0 & 0 & 0.0001 \end{bmatrix}
\end{aligned}
\tag{1.18}
$$

The link inertia matrices are expressed in the reference system of the i-th link identified by the DH parameters, and the unit of measure is $Kg \cdot m^2$.

1.5 Control

The trajectories generated constitute the reference inputs to the motion control system of the mechanical structure. The problem of robot manipulator control is to find the time behaviour of the forces and torques to be delivered by the joint actuators so as to ensure the execution of the reference trajectories. Several techniques can be employed for controlling a manipulator. These techniques can affect the performance of the manipulator. The *joint space control problem* is actually articulated in two subproblems. First, manipulator inverse kinematics is solved to transform the motion requirements $\mathbf{x}_d$ from the operational space into the corresponding motion $\mathbf{q}_d$ in the joint space. Then, a joint space control scheme is designed that allows the actual motion $\mathbf{q}$ to track the reference inputs.

1.5.1 Joint Space control

The equations of motion of a manipulator in the absence of external end-effector forces and, for simplicity, of static friction (difficult to model accurately) are described by:

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \tau \tag{1.19}$$

To control the motion of the manipulator in free space means to determine the n components of generalized forces - torques for revolute joints, forces for prismatic joints - that allow execution of a motion $\mathbf{q}(t)$ so that

$$\mathbf{q}(t) = \mathbf{q}_d(t) \tag{1.20}$$

as closely as possible, where $\mathbf{q}_d(t)$ denotes the vector of desired joint trajectory variables [6].

1.5.2 PD Control with Gravity compensation

The simplest control strategy that can be thought of is one that regards the manipulator as formed by n independent systems (the n joints) and controls each joint axis as a single-input/single-output system. Coupling effects between joints due to varying configurations during motion are treated as disturbance inputs (*Decentralized control*). On the other hand, when large operational speeds are required or direct-drive actuation is employed, the nonlinear coupling terms strongly influence system performance. In this case, it is advisable to design control algorithms that take advantage of a detailed knowledge of manipulator dynamics so as to compensate for the

nonlinear coupling terms of the model. In other words, it is necessary to eliminate the causes rather than to reduce the effects induced by them; that is, to generate compensating torques for the nonlinear terms. This leads to *centralized control* algorithms that are based on the (partial or complete) knowledge of the manipulator dynamic model. As shown by the dynamic model 1.19, the manipulator is not a set of n decoupled system but it is a multivariable system with *n inputs (joint torques) and n outputs (joint positions)* interacting between them by means of nonlinear relations.

Let a constant equilibrium posture be assigned for the system as the vector of desired joint variables $\mathbf{q}_d$. It is desired to find the structure of the controller which ensures global asymptotic stability of the above posture. *The determination of the control input which stabilizes the system around the equilibrium posture is based on the Lyapunov direct method.* Take the vector $[\tilde{\mathbf{q}}^T \ \ \dot{\mathbf{q}}^T]^T$ as the system state, where:

$$\tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q} \tag{1.21}$$

represent the error between the desired and the actual posture. Choose the following positive definite quadratic form as Lyapunov function candidate:

$$V(\dot{\mathbf{q}}, \tilde{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{B}(\mathbf{q})\dot{\mathbf{q}} + \frac{1}{2}\tilde{\mathbf{q}}^T\mathbf{K}_P\tilde{\mathbf{q}} > 0 \quad \forall\tilde{\mathbf{q}}, \dot{\mathbf{q}} \neq 0 \tag{1.22}$$

where $\mathbf{K}_p$ is an $(n \times n)$ symmetric positive definite matrix. An energy-based interpretation of 1.22 reveals a first term expressing the system kinetic energy and a second term expressing the potential energy stored in the system of equivalent stiffness $\mathbf{K}_P$ provided by the n position feedback loops. Differentiating 1.22 with respect to time, and recalling that $\mathbf{q}_d$ is constant, yields

$$\dot{V} = \dot{\mathbf{q}}^T\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \frac{1}{2}\dot{\mathbf{q}}^T\dot{\mathbf{B}}(\mathbf{q})\dot{\mathbf{q}} - \dot{\mathbf{q}}^T\mathbf{K}_P\tilde{\mathbf{q}} \tag{1.23}$$

Solving 1.19 for $\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}}$ and substituting it in 1.23 gives

$$\dot{V} = \frac{1}{2}\dot{\mathbf{q}}^T(\dot{\mathbf{B}}(\mathbf{q}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}))\,\dot{\mathbf{q}} - \dot{\mathbf{q}}^T\mathbf{F}\dot{\mathbf{q}} + \dot{\mathbf{q}}^T(\mathbf{u} - \mathbf{g}(\mathbf{q}) - \mathbf{K}_P\tilde{\mathbf{q}}) \tag{1.24}$$

The first term of the equation is zero due to Christoffel symbols.

$$\mathbf{N} = \dot{\mathbf{B}} - 2\mathbf{C} \tag{1.25}$$

The second term is negative definite. Then, the choice

$$\mathbf{u} = \mathbf{g}(\mathbf{q}) + \mathbf{K}_P\tilde{\mathbf{q}} \tag{1.26}$$

describing a controller with compensation of gravitational terms and a proportional action, leads to a negative semi-definite $\dot{V}$ since

$$\dot{V} = 0 \quad \dot{\mathbf{q}} = \mathbf{0}, \forall\tilde{\mathbf{q}}$$

. This result can be obtained also by taking the control law:

$$\mathbf{u} = \mathbf{g}(\mathbf{q}) + \mathbf{K}_P\tilde{\mathbf{q}} - \mathbf{K}_D\dot{\mathbf{q}} \tag{1.27}$$

with $\mathbf{K}_D$ positive definite, corresponding to a nonlinear compensation action of gravitational terms with a linear proportional-derivative (PD) action. In fact, substituting 1.27 into 1.23 gives

$$\dot{V} = -\dot{\mathbf{q}}^T\left(\mathbf{F} + \mathbf{K}_D\right)\dot{\mathbf{q}}, \tag{1.28}$$

which reveals that the introduction of the derivative term causes an increase of the absolute values of $\dot{V}$ along the system trajectories, and then it gives an improvement of system time response. According to the above, the function candidate V decreases as long as $\dot{\mathbf{q}} \neq 0$ for all system trajectories. It can be shown that the system reaches an equilibrium posture. To find such posture, notice that $\dot{V} = 0$ only if $\dot{q} = 0$. System dynamics under control 1.27 is given by

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{g}(\mathbf{q}) + \mathbf{K}_P\tilde{\mathbf{q}} - \mathbf{K}_D\dot{\mathbf{q}} \tag{1.29}$$

At the equilibrium ($\dot{\mathbf{q}} = 0, \ddot{\mathbf{q}} = 0$) it is

$$\mathbf{K}_P\tilde{\mathbf{q}} = \mathbf{0} \tag{1.30}$$

and then

$$\tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q} = \mathbf{0} \tag{1.31}$$

is the sought equilibrium posture. The above derivation rigorously shows that any manipulator equilibrium posture is globally asymptotically stable under a controller with a PD linear action and a nonlinear gravity compensating action. Stability is ensured for any choice of $\mathbf{K}_P$ and $\mathbf{K}_D$, as long as these are positive definite matrices. The resulting block scheme is shown in Fig.1.10.

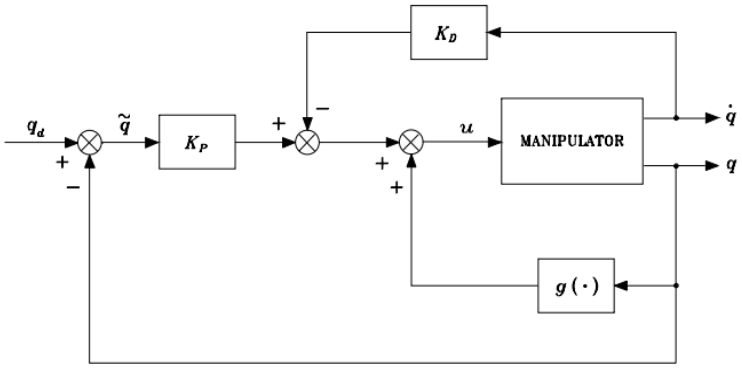


Figure 1.10: Block scheme of joint space PD control with gravity compensation.

The control law requires the on-line computation of the term $\mathbf{g}(\mathbf{q})$. If compensation is imperfect, the above discussion does not lead to the same result; this aspect will be revisited later with reference to robustness of controllers performing nonlinear compensation [6].

Gravity Compensation

The essential part of the controller is gravity compensator, because gravity term is generally the dominant term in robot dynamics. It presents even when the robot is stationary or moving slowly. To calculate the gravity term, potential energy for each link has to be calculated which is configuration dependent. Below the formula illustrates the calculation of potential energy for each link:

$$P_i = m_i g^T r_{ci} \tag{1.32}$$

where r_{ci} is the coordinate of the center of mass of link i. After getting each link's potential energy, the overall potential energy of the system has to be calculated by summing the individual potential energies which is given below (ignoring robot elasticity):

$$P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} m_i g^T r_{ci} \tag{1.33}$$

To obtain the gravity term for each of the manipulator's link, the partial derivative of the total potential energy has to be calculated with respect to each of the joint generalized coordinates as the following:

$$g_k = \frac{\partial P}{\partial q_k} \tag{1.34}$$

To implement the gravity compensator the Peter Corke robotics Toolbox function *gravload()* is used.

Chapter 2

Harmonic Drive

In general the transmission elements of robot drive system should have important features, such as zero backlash, low weight, low friction losses, compact size and high torque capacity. These features strongly influence manipulator performance. For this reason, it is important to study the dominant performance characteristics of robot transmissions.

Harmonic drives Fig 2.1, invented in the 1950s, are widely used in robotic systems, due to their desirable features of near zero backlash, compactness, light weight, high torque capacity, high gear ratio and coaxial assembly. These distinctive characteristics of harmonic drives vindicate their widespread applications, especially in electrically-driven robot manipulators. An Harmonic drive reduces the gear ratio of a rotary machine to increase torque. It operates on a principle different from that of conventional speed changers. The device consists of a thin ring that deflects elastically as it rolls inside a slightly larger rigid circular ring. It is used to achieve precise and compact motion control. More in details, it consists of three main components:

- **Wave Generator** (WG): The Wave Generator is a thin, race-ball bearing fitted onto an elliptical hub. This serves as high efficiency torque converter and is generally mounted onto the input or motor shaft.
- **Circular Spline** (CS): The Circular Spline is rigid ring with internal teeth. It engages the teeth of the Flexspline across the major axis of the Wave Generator ellipse. The circular spline has two more teeth than the Flex spline and is generally mounted onto a housing.
- **Flexspline** (FS): The Flexspline is a non-rigid (flexible) thin cylindrical cup with external teeth on the open end of the cup. The Flexspline fits over the Wave Generator and takes on its elliptical shape. In particular connects the Wave Generator to the Circular Spline. The Flex spline is generally used as the output of the gear.

Typically the WG is connected to the motor shaft, the CS is connected to the joint housing, and the FS is sandwiched in between (CS and WG) and connected to the joint output. The WG consists of an elliptical disk (rigid elliptical inner-race), called wave generator plug, and an outer ball bearing. The wave generator plug is inserted into the bearing, thereby giving the bearing an elliptical shape as well. The FS fits tightly over WG; when the WG plug is rotated, the FS deforms and molds into the shape of the rotating ellipse but does not rotate with WG.

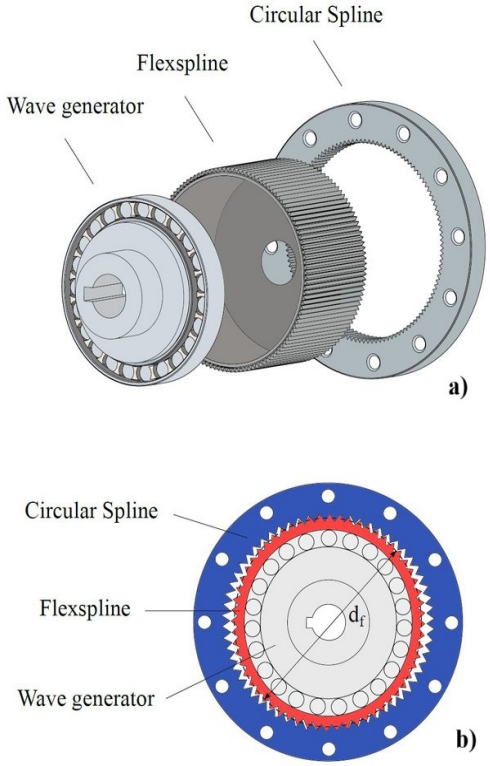


Figure 2.1: a) Harmonic drive gearbox; b) Harmonic drive cross section [4]

The principle of a harmonic drive involves the elastic deformation of the Flexspline to produce a high gear reduction ratio in a compact space. The wave generator is connected to the input, and as it rotates, it causes the flex spline to flex and engage with the circular spline. This interaction generates an output rotation at a reduced speed compared to the input, achieving a high reduction ratio. The ratio of the input speed to the output speed depends on the difference in the number of teeth in the circular spline and on the Flexspline.

The schematic view of a harmonic drive in four operating positions is shown in Fig.2.2. The ball bearing mounted on the elliptical wave gener-

ator deforms the flexspline, thus engaging teeth at diametrically opposite points coincident with the major axis of the elliptical wave generator and disengaging points at the minor axis (Fig.2.2(a)). When the circular spline is fixed, the rotation of the wave generator produces a reverse motion of the flexspline (Fig.2.2 (b), (c)). If the rigid circular spline has two teeth more than the flexspline, then a single revolution of the wave generator rotates the flexspline backward about two teeth (Fig.2.2(d)). Thus, the speed of the wave generator is reduced approximately $N_t/2$ times, where N_t is the number of teeth on the Flexspline.

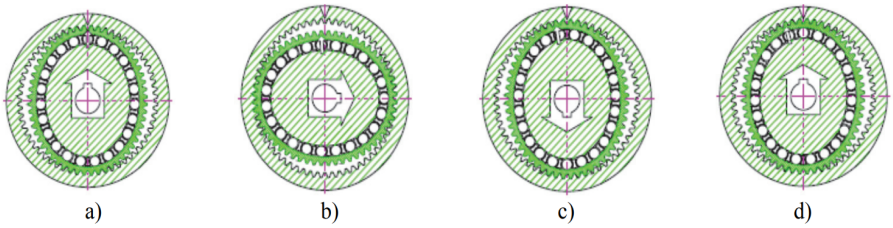


Figure 2.2: The operating positions of a harmonic drive: a) the elliptical shape of the wave generator causes the teeth of the flexspline to engage the circular spline at two regions at opposite ends of the major axis of the ellipse, b) as the wave generator rotates the zone of tooth engagement travels with the major axis of the ellipse, c) for each 180° clockwise movement of the wave generator the flexspline moves counterclockwise by one tooth relative to the circular spline, d) each complete clockwise rotation of the wave generator results in the flexspline moving counterclockwise by two teeth from its previous position relative to the circular spline

Harmonic drives are not the only choice for actuation in robotic joints like the ones found in the UR5. There are several other types of actuators and transmission mechanisms that can be used, depending on the specific requirements of the robot and its intended application. Here are some alternative options: Traditional Gearboxes, Direct Drive Motors, Pneumatic or Hydraulic Actuators ecc. In our case the use of Harmonic drives helps to ensure precise, reliable, and efficient operation of the robotic arm in various industrial and research applications.

2.1 Mathematical Model

Harmonic drives are a type of gear mechanism that are very popular for use in robots due to their low backlash, high torque transmission and compact size. However, *they also introduce unwanted friction and flexibility at the joints* [7]. For many manipulators, particularly those using harmonic drives, for torque transmission, the joint flexibility is significant. In general, in

addition to torsional flexibility in the gears, joint flexibility is caused by effects such as shaft windup, bearing deformation, and compressibility of the hydraulic fluid in hydraulic robots.

2.1.1 Kinematic

As explained previously, the flexspline has two fewer teeth than the circular spline, and thus each full turn of the wave generator moves the flexspline two teeth in the opposite direction relative to the circular spline [2]. This is a Kinematic constraint given by:

$$q_w = Nq_f + (N+1)q_c \tag{2.1}$$

where N is the gear ratio defined as $N = N_t/2$, and N_t is the number of teeth on the flexspline outer circumference. Circular spline will be considered as fixed, so q_c will be zero which will simplify the above equation as follows:

$$q_w = Nq_f \tag{2.2}$$

in derivative form:

$$\dot{q_w} = N\dot{q_f} \tag{2.3}$$

The static force balance can be described as:

$$\tau_w = \frac{1}{N}\tau_f \tag{2.4}$$

where τ_w is the torque at the wave generator and τ_f is the flexspline output torque. The previous two equations represent the harmonic drivés ideal linear input/output relationship in which the harmonic drive transmission is treated as a perfectly rigid gear reduction mechanism. However, often *in real applications the output is not linearly related to the input. The causes of this nonlinearities are torsional compliance in the harmonic drive components, nonlinear friction forces, and the kinematic error that is due to gear meshing and machining errors.* Given this ideal kinematic relationship which describes the motion and force constraints present in harmonic drives, the remaining effects can be incorporated by modeling compliance, friction and kinematic error [5].

2.1.2 Harmonic drive nonlinear friction

Harmonic drive model contains friction terms due to the internal elements of the harmonic drive. All harmonic drives exhibit power loss during operation due to transmission friction. The bulk of energy dissipation can be blamed on the wave generator bearing friction, gear meshing friction, output bearing friction and flexspline structural damping. Among them, most of the frictional dissipation results from gear meshing [8].

We need to consider these friction effects. These effects usually appear only at very low velocities. For this reason, we can use the following model:

$$\tau_b = B\dot{q}_w + (B^+, B^-)sgn(\dot{q}_w) \tag{2.5}$$

The friction term is nonlinear and change depending on the sign of the wave generator's velocity. The above equation is used in Simscape to build a custom nonlinear damper. The code is shown in section 2.2.

2.1.3 Harmonic drive compliance model

The compliance behavior of the harmonic drive is illustrated in Fig.2.3, with consideration of the flexspline and wave generator compliance.

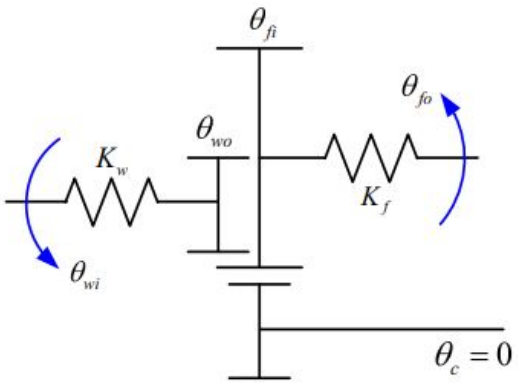


Figure 2.3: Kinematic representation of a harmonic drive [5].

A typical harmonic drive stiffness curve is shown in Fig.2.4, which features increasing stiffness with displacement and hysteresis behavior. The total harmonic drive torsional deformation comprises deformation of both the flexspline and the wave generator [5]. Based on experimental observation, *the harmonic drive torsional deformation is largely contributed by the flexspline torsional compliance.* We can approximate the $\Delta\theta_f$ with a piecewise linear function of the output torque Fig.2.4(b).

$$\Delta\theta_f = \begin{cases} \frac{\tau_f}{K_1}, & \tau_f \leq \tau_1 \\ \frac{\tau_1}{K_1} + \frac{\tau_f - \tau_1}{K_2}, & \tau_1 < \tau_f < \tau_2 \\ \frac{\tau_1}{K_1} + \frac{\tau_f - \tau_1}{K_2} + \frac{\tau_f - \tau_2}{K_3}, & \tau_f \geq \tau_2 \end{cases} \tag{2.6}$$

where K_1, K_2, K_3, τ_1, and τ_2 are given by the manufacture. *The slope of the curve shown in Fig. 2.4 indicates the harmonic drive stiffness.* The curve is approximated by three straight-line segments with stiffness of K_1, K_2 and K_3. Stiffness K_1 applies for flexspline torque of 0 to τ_1; stiffness K_2 applies

for flexspline torque ranging from τ_1 to τ_2; and stiffness K_3 applies for flexspline torque greater than τ_2.

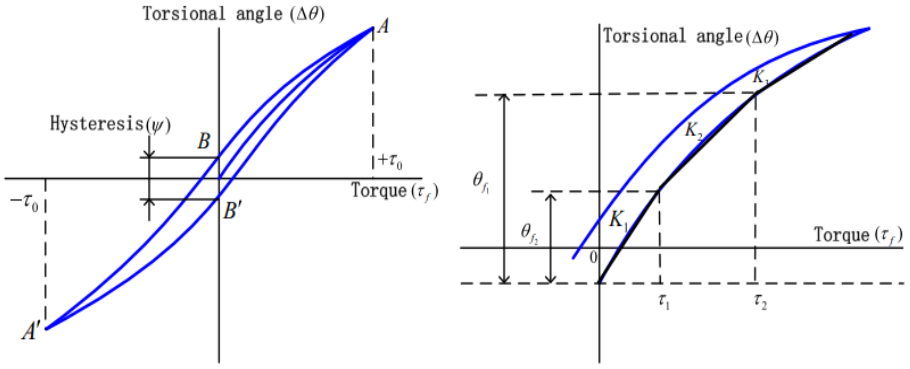


Figure 2.4: Typical stiffness and hysteresis curve of a harmonic drive.

Using the inverse of Eq.2.6, the flexspline output torque τ_f, the joint torque to be estimated, can be presented as:

$$\tau_f = \begin{cases} K_1 \Delta\theta_f, & \Delta\theta_f \leq \theta_{f_1} \\ \tau_1 + K_2(\Delta\theta_f - \theta_{f_1}), & \theta_{f_1} < \Delta\theta_f < \theta_{f_2} \\ \tau_2 + K_3(\Delta\theta_f - \theta_{f_2}), & \Delta\theta_f \leq \theta_{f_2} \end{cases} \tag{2.7}$$

where $\theta_{f_1} = \tau_1/K_1$ and $\theta_{f_2} = \tau_1/K_1 + (\tau_2 - \tau_1)/K_2$. The above equations are used in Simscape to build a custom nonlinear spring (code notation: in the code we indicate $\Delta\theta_f$ as fi). The code is shown in section 2.2.

2.1.4 Force distribution

For a complete understanding of the operation of the harmonic drive it is necessary to consider the geometry of the teeth and the interaction forces. To explain the force transmission mechanism within the harmonic transmission we can consider Fig.2.5. Since the meshing in the top and bottom halves of the harmonic drive is symmetric, just the top half needs to be examined. When looking directly at the center of the harmonic drive, it can be seen that the meshing is once again symmetric and mirrored between the left and the right halves. Thus, *in the top-right half, the right face of each contacting gear tooth on the flexspline is in contact with the left face of each contacting gear tooth on the circular spline.* The determination of the geometry of meshing is critical in analyzing the torque relationships [2]. Suppose that the circular spline is fixed to ground, and that the flexspline is loaded by a rotational spring. The clockwise rotation of the wave generator creates a load torque on the wave generator that opposes the driving torque, τ_w.

The existence of the load torque can be explained as follows. The circular spline reaction force f_i, which is normal to the tooth i, acts on the flexspline tooth being engaged with the tooth i. Due to the elliptic shape of the wave generator, the force f_i has a normal component to the ellipse that creates the torque component $f_i cos(\Phi - \mu_i)\rho_i$, which acts against the driving torque, τ_w. Here, Φ is the gear-tooth profile angle, μ_i is the angle between the normal to the circular spline and the normal to the flexspline at the contact point (tooth i in Figure 2.5), and ρ_i is the distance from the center of rotation to the normal to the ellipse at the same point [2]. Taking into account all teeth being engaged, and integrating all torque components of the reaction torque, it can be shown that the following relationship holds:

$$\tau_f = N\tau_w \tag{2.8}$$

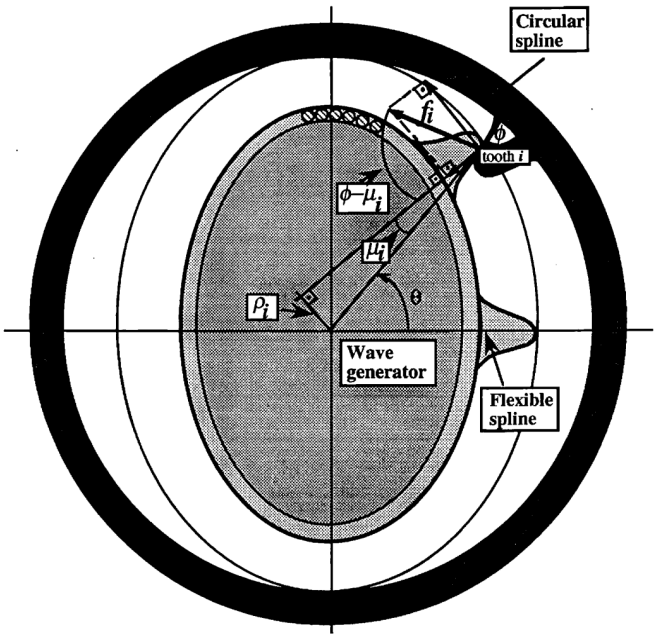


Figure 2.5: Force distribution [2]

2.2 Simscape

In order to implement the model of Harmonic Drive in Simscape, simplified bond graph of the system is given in figure 2.6:

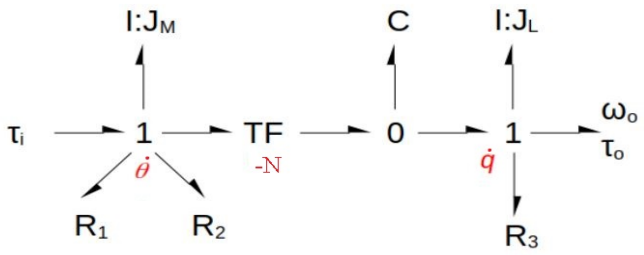


Figure 2.6: Harmonic Drive bond graph

In UR5 robot manipulator two different harmonic drive modules have been utilized, namely for the first three joints **HFUS-25** and for the last three joints **HFUS-14**, with gear ration of $N = 100$. Based on the bond graph shown in Figure 2.6, the harmonic driving model can be found. This model will create for each joint of the robot manipulator. The input to the subsystem is torque from the controller.

	Symbol [Unit]	HFUS-25					
Ratio	i [ ]	30	50	80	100	120	160
Repeated peak toque	T_R [Nm]	50	98	137	157	167	176
Average torque	T_A [Nm]	38	55	87	108	108	108
Rated torque	T_N [Nm]	27	39	63	67	67	67
Momentary peak torque	T_M [Nm]	95	186	255	284	304	314
Maximum input speed (oil lubrication)	$n_{in\,(max)}$ [rpm]	7500					
Maximum input speed (grease lubrication)	$n_{in\,(max)}$ [rpm]	5600					
Average input speed (oil lubrication)	$n_{av\,(max)}$ [rpm]	5600 / 1000[1]					
Average input speed (grease lubrication)	$n_{av\,(max)}$ [rpm]	3500 / 1000[1]					
Moment of inertia HFUS-2UH	J_m [$\cdot 10^{-4}$ kgm^2]	1.07					
Moment of inertia HFUS-2SO	J_m [$\cdot 10^{-4}$ kgm^2]	0.413					
Moment of inertia HFUS-2SH	J_m [$\cdot 10^{-4}$ kgm^2]	1.07					
Weight HFUS-2UH	m [kg]	2.1					
Weight HFUS-2SO	m [kg]	1.31					
Weight HFUS-2SH	m [kg]	1.44					

Figure 2.7: HFUS-25 General Technical Data

	Symbol [Unit]	HFUS-25			HFUS-32			HFUS-40	
Limit torque	T_1 [Nm]	14			29			54	
	T_2 [Nm]	48			108			196	
Ratio	i [ ]	**30**	**50**	**≥ 80**	**30**	**50**	**≥ 80**	**50**	**≥ 80**
Torsional Stiffness	K_3 [$\cdot 10^3$ Nm/rad]	21	44	57	49	98	120	180	230
	K_2 [$\cdot 10^3$ Nm/rad]	13	34	50	30	78	110	140	200
	K_1 [$\cdot 10^3$ Nm/rad]	10	25	31	24	54	67	100	130

Figure 2.8: HFUS-25 Torsional spring data

	Symbol [Unit]	HFUS-14			
Ratio	i [ ]	30	50	80	100
Repeated peak toque	T_R [Nm]	9.0	18	23	28
Average torque	T_A [Nm]	6.8	6.9	11	11
Rated torque	T_N [Nm]	4.0	5.4	7.8	7.8
Momentary peak torque	T_M [Nm]	17	35	47	54
Maximum input speed (oil lubrication)	$n_{in\,(max)}$ [rpm]	14000			
Maximum input speed (grease lubrication)	$n_{in\,(max)}$ [rpm]	8500			
Average input speed (oil lubrication)	$n_{av\,(max)}$ [rpm]	6500 / 1100[1)]			
Average input speed (grease lubrication)	$n_{av\,(max)}$ [rpm]	3500 / 1100[1)]			
Moment of inertia HFUS-2UH	J_{in} [$\cdot 10^{-4}$ kgm²]	0.091			
Moment of inertia HFUS-2SO	J_{in} [$\cdot 10^{-4}$ kgm²]	0.033			
Moment of inertia HFUS-2SH	J_{in} [$\cdot 10^{-4}$ kgm²]	0.091			
Weight HFUS-2UH	m [kg]	0.71			
Weight HFUS-2SO	m [kg]	0.41			
Weight HFUS-2SH	m [kg]	0.45			

Figure 2.9: HFUS-14 General Technical Data

	Symbol [Unit]	HFUS-14			HFUS-17			HFUS-20		
Limit torque	T_1 [Nm]	2			3.9			7		
	T_2 [Nm]	6.9			12			25		
Ratio	i [ ]	**30**	**50**	**≥ 80**	**30**	**50**	**≥ 80**	**30**	**50**	**≥ 80**
Torsional Stiffness	K_3 [$\cdot 10^3$ Nm/rad]	3.4	5.7	7.1	6.7	13	16	11	23	29
	K_2 [$\cdot 10^3$ Nm/rad]	2.4	4.7	6.1	4.4	11	14	7.1	18	25
	K_1 [$\cdot 10^3$ Nm/rad]	1.9	3.4	4.7	3.4	8.1	10	5.7	13	16

Figure 2.10: HFUS-14 Torsional stiffness

The components used to build the harmonic driving model are:

- **Mechanical Rotational Reference** this block is used to define a fixed reference point for measuring rotation.
- **Ideal Torque Source Block**: This block represents an ideal source of mechanical energy that generate torque proportional to the input physical signal. To represent the power input to the system.
- **Inertia**: This block represent an ideal mechanical rotational inertia that is described with the following equation:

$$T = J\frac{d\omega}{dt} \tag{2.9}$$

  where T is the Torque, J is the inertia, ω is the angular velocity and t is the time.

- **Simple Gear Block (Ideal)**: To model the reduction ratio that is characteristic of Harmonic Drives. This block represent a gearbox thet contains the connected driveline axes of the *basic gear*, B, and the *follower gear*, F, to corotate with a fixed ratio N. The kinematic constraint that it imposes on the two connected axes is:

$$r_F\omega_F = r_B\omega_B \tag{2.10}$$

the *follower - base gear ratio* is:

$$N = \frac{r_F}{r_B} = \frac{N_F}{N_B} \tag{2.11}$$

where N_F and N_B are the number of teeth in the follower gear and in the base gear,respectively. Furthermore, the following relation hold:

$$N\tau_B + \tau_F = 0 \tag{2.12}$$

without considering the loss.

- **Nonlinear Spring Block**: To represent the compliance of the Harmonic Drive, which can be significant due to the elastic deformation of the drive elements. The following is the Matlab code inside the block:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The nonlinearSpring component models a spring
   with nonlinear behavior, where the torque
   depends on the deformation angle and the
   stiffness constants.

component nonlinearSpring

% The component has two nodes, labeled is (left)
    or (right). These represent the ends of the
   spring.
nodes
    r = foundation.mechanical.rotational.
       rotational; % r:left
    c = foundation.mechanical.rotational.
       rotational; % c:right
end

parameters
    K1 = {31e3, 'N*m/rad' };
    K2 = {50e3, 'N*m/rad' };
    K3 = {57e3, 'N*m/rad' };
    T1 = {14, 'N*m'};
    T2 = {48, 'N*m'};
end
% K1, K2, and K3 are the spring stiffness
   constants in terms of N*m/rad. T1 and T2 are
   the torques applied to the spring in terms of
    N*m.
```

```
parameters (Access = private)
    fi1 = T1 / K1;
    fi2 = T1 / K1 + (T2 - T1) / K2;
end

variables
    theta = { 0, 'rad' };
    t = { 0, 'N*m' };
    w = { 0, 'rad/s' };
end
% theta represents the spring deformation angle
   in radians. t represents the torque on the
   spring in terms of N*m. w represents the
   angular velocity of the spring in rad/s.

branches
    t : r.t -> c.t;
end

equations
    assert(K1 >= 0 & K2 >=0 & K3 >= 0, '
       Stiffness must be >= 0');

    w == r.w - c.w;
    % The equation w == r.w - c.w dictates that
       the angular velocity of the spring is the
        difference between the angular
       velocities of the left and right nodes.

    w == theta.der;
    % The previous equation connects the angular
        velocity to the derivative of the
       deformation angle.

    if theta <= fi1 && theta >= -fi1
        t == theta * K1;
    elseif theta > fi2 || theta < -fi2
        t == (theta - sign(theta) * fi2) * K3 +
           sign(theta) * T2;
    else
        t == (theta - sign(theta) * fi1) * K2 +
           sign(theta) * T1;
    end
    % The equation for torque t depends on the
```

```
        range of deformation angles: If theta is
        between -fi1 and fi1, the torque is
        proportional to the deformation angle
        multiplied by K1. If theta is outside
        this range, the torque is calculated
        based on a combination of constants and
        specific torques.
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

- **Nonlinear Damper Block**: To account for damping due to interactions between moving parts. The following is the cose inside the block:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
component nonlinearDamper

nodes
    r = foundation.mechanical.rotational.
       rotational;%R:left
    c = foundation.mechanical.rotational.
       rotational;%C:right
end

parameters  % Damping coefficients
    B = {0.00064, 'N*m/(rad/s)'};
    B1 = {0.008,'N*m'};
    B2 = {-0.007,'N*m'};
end

variables
 t = {0,'N*m'};
 w = {0, 'rad/s'};
end

branches
    t:r.t -> c.t;
end

% The following equations describe the behavior
   of the nonlinear damper:

equations
```

```
        % If angular velocity w is positive:
        if w>0
        t == B*w+B1;
        else
        t == B*w-B2;
        end
        w == r.w - c.w;

    end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

- **Ideal rotational motion sensor Block**: to monitor the angular position and speed on both the motor and load sides.
- **Rotational Multibody Interface Block**: this block implement an interface between mechanical rotational networks and simscape multibody joint. In particular, it is used to connect the Harmonic Drive model to the rest of the robot model. This block is necessary to modelling nonlinear spring, dampers, friction for Simscape multibody joints.
- **Connections and control signals**: the green connections are the physical mechanical connections, while the brown lines are the control or measurement signals.

The *Nonlinear Spring* and *Nonlinear Damper* components simulate the elastic nature and dissipative losses within the Harmonic Drive.

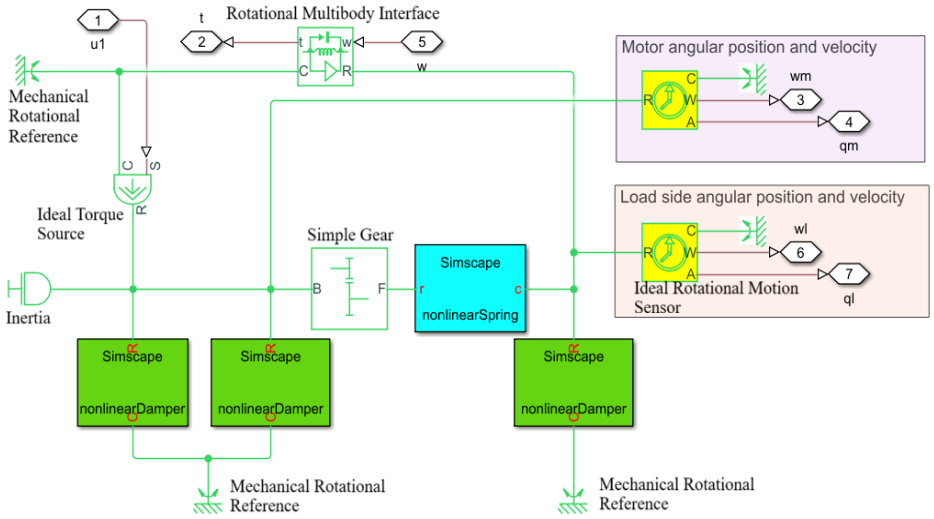


Figure 2.11: Harmonic Drive Simscape Model

2.3 Controller

As mentioned previously, PD Control with Gravity Compensation is used to control the robot manipulator to track the desired trajectory. However, after adding the Harmonic Driving model to the overall robot model, should also be taken into account to successfully track the given trajectory. In this project, the *LQR controller* will be applied to achieve the desired motion.

2.3.1 LQR Controller

The Linear Quadratic Regulator (LQR), Fig.2.12 is a full state feedback optimal control law, $\mathbf{u} = -\mathbf{K}\mathbf{x}$, that minimizes a quadratic cost function to regulate the control system. The quadratic cost function that takes into account both the state error and the magnitude of the control command. The cost function (for continuous time system) is defined as follows:

$$J = \int (\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u} + 2\mathbf{x}^T\mathbf{N}\mathbf{u})dt \tag{2.13}$$

where:

- $\mathbf{x}$: state of the system
- $\mathbf{u}$: control command
- $\mathbf{Q}$: weight matrix for the state
- $\mathbf{R}$: weight matrix for the control command

- **N**: Optional cross-term matrix

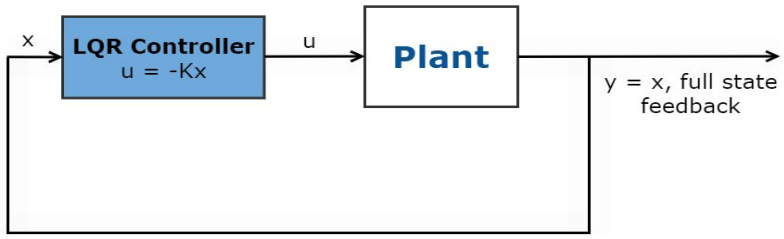


Figure 2.12: General Schematic of Linear Quadratic Regulator controller

The **Q** and **R** matrices are used to tune the performance of the controller. The **Q** matrix assigns a relative weight to the different states of the system. *Each element on the diagonal of **Q** corresponds to how much weight to give to the error associated with a specific state of the system.* Dimensionally, **Q** is an $n \times n$ matrix, where n is the dimension of the system's state vector. *Larger values in **Q** indicate that we want the control to be more robust to that aspect of the system.* **R** is a weight matrix that assigns a relative weight to the control action. *Each element on the diagonal of **R** indicates how much weight to give to the control associated with a particular system input.* Dimensionally, **R** is an $m \times m$ matrix, where m is the dimension of the control input vector. *Larger values in **R** indicate that we want to minimize variation in controls.* The LQR controller is used in the control scheme because it offers several advantages over other types of controllers:

- Stability: it guarantees system stability if the Q and R matrices are defined correctly.
- Performance: it can be tuned to achieve the desired performance in terms of precision, speed and robustness.
- Robustness: it is robust to disturbances and measurement noise.
- Ease of Implementation: it is easy to implement in software and hardware.

2.3.2 LQR Design

However, to build the LQR controller, the differential equations representing the system must be obtained. In order to obtain these equations we can start considering the fully augmented graph 2.13

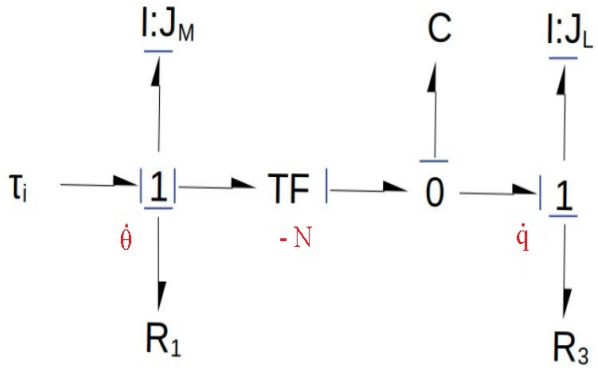


Figure 2.13: Bond graph with causalities

We can observe that there is neither algebraic loop nor derivative causality. We have three store energy components so we will have three differential equations. The state vector is:

$$x = \begin{bmatrix} p_m \\ p_l \\ \phi \end{bmatrix} \tag{2.14}$$

The bond graph equations can be written in terms of state elements:

$$\dot{p}_m = \tau_i - \frac{K\phi}{N} - \frac{Bp_m}{J_m} \tag{2.15}$$

$$\dot{\phi} = -\frac{p_l}{J_l} + \frac{p_m}{J_m N} \tag{2.16}$$

$$\dot{p}_l = K\phi - \frac{Bp_l}{J_l} \tag{2.17}$$

ϕ can be obtained from equation 2.16 by integrating both sides. If the obtained value of ϕ is plugged into equations 2.15 and 2.17, then below differential equations can be obtained:

$$J_m\ddot{\theta} + B\dot{\theta} + \frac{K}{N}(\frac{\theta}{N} - q) = \tau \tag{2.18}$$

$$J_l\ddot{q} + B\dot{q} + K(q - \frac{\theta}{N}) = 0 \tag{2.19}$$

Where:

- J_m: inertia motor side
- J_l: inertia load side

- τ_i: input torque
- N: gear ratio
- B: damping coefficient
- K: spring constant
- $\dot{\theta}$: the motor side angular velocity
- $\dot{q}$: the load side angular velocity

Since the LQR controller is designed based on linear model, nonlinearities due to the damping and spring elements have been ignored. In the next step, state-space model of the system has to be obtained. In order to do that the system equations can be rewritten by choosing the state variables as:

$$\begin{array}{ll} x_1 = q & x_3 = \theta \\ x_2 = \dot{q} & x_4 = \dot{\theta} \end{array} \tag{2.20}$$

So the system equations become:

$$\dot{x_1} = x_2 \tag{2.21}$$

$$\dot{x_2} = -\frac{B}{J_l}x_2 - \frac{K}{J_l}x_1 + \frac{K}{J_l N}x_3 \tag{2.22}$$

$$\dot{x_3} = x_4 \tag{2.23}$$

$$\dot{x_4} = \frac{\tau}{J_m} - \frac{B}{J_m}x_4 - \frac{K}{J_m N^2}x_3 + \frac{K}{J_m N}x_1 \tag{2.24}$$

These equations can be grouped in matrix form as below:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{K}{J_l} & -\frac{B}{J_l} & \frac{K}{J_l N} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{K}{J_m N} & 0 & -\frac{K}{J_m N^2} & -\frac{B}{J_m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J_m} \end{bmatrix} \begin{bmatrix} \tau \end{bmatrix} \tag{2.25}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{2.26}$$

From above matrix form equations, *state-space* matrices (A, B and C) can be extracted easily. After the state-space matrices have been obtained, a *linear state feedback controller* can be written as:

$$u(t) = -K^T x + u_r \tag{2.27}$$

where, u_r is the reference input which is the output from PD with gravity compensation controller and K is the *optimal feedback gain* which is obtained by solving algebraic *Riccati equation.*

Chapter 3

Simulation

Now all the main results of the simulation of the **ur5ControlScheme.slx** file are reported. As a first step we will consider the UR5 + Harmonic Drive model then the control part and finally the joint/workspace trajectories.

3.1 Complete Control Scheme

As shown in Fig.3.1, the overall control scheme consists of six blocks:

- UR5_with_HD block
- PID_with_Gravity_Compensation block
- LQR Controller block
- Trajectories block
- Joint and Workspace Trajectory block

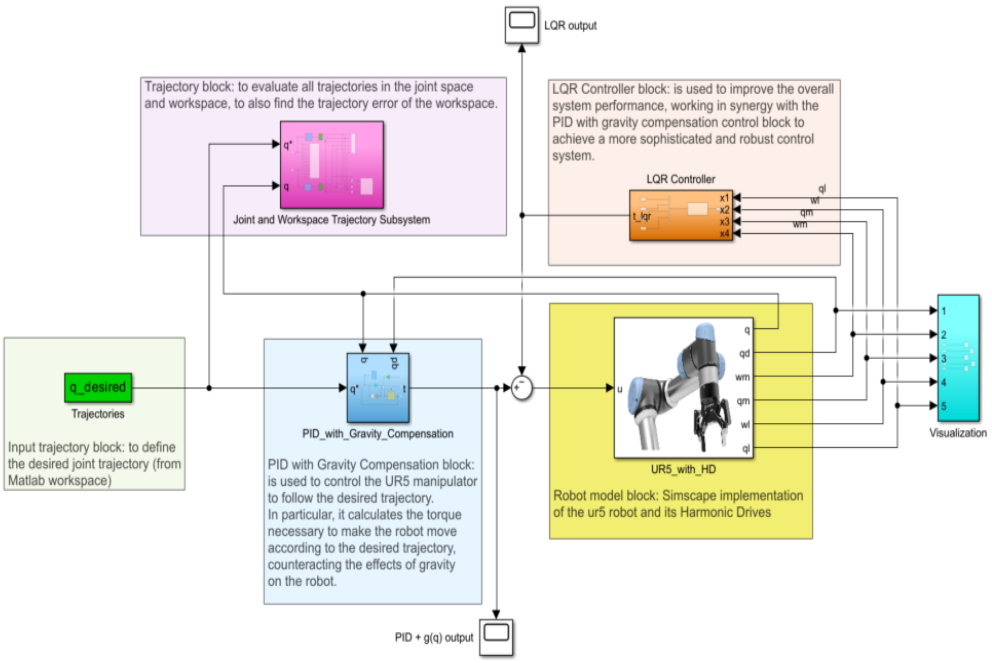


Figure 3.1: UR5 total control scheme

In the following sections the various results are reported considering *two different final configurations* (therefore considering two different joint/-workspace trajectories), explained in the previous section 1.2.2.

3.1.1 UR5 + Harmonic Drive Simscape model

Simscape Multibody provides a multibody simulation environment for 3D mechanical systems, such as robots. Whit this Matlab extension we import CAD geometries, masses, inertias, joints, constraints, and 3D geometries from PTC creo into our model. The final model is presented in figure 3.2.

From the figure 3.2 it can be seen that the *actuation torque* (t revolute joint input) to the revolute joint i is given by the harmonic drive i. The Harmonic drive i takes as input the control input u_i and the output (w) angular velocity from the revolute joint i; with $i = 1 \dots 6$. The control input comes from the control block, in particular it is given by the difference between LQR and PD with gravity compensator. In addition to the torque (t), the other outputs of the harmonic drive i are ω_{m_i}, q_{m_i} that represent the angular velocity and position at the motors side and ω_{l_i}, q_{l_i} that represent the angular velocity and position at the load side. We use all these output as input for the LQR block. Running the model 3.2, we obtain the result represented in figure 3.3.

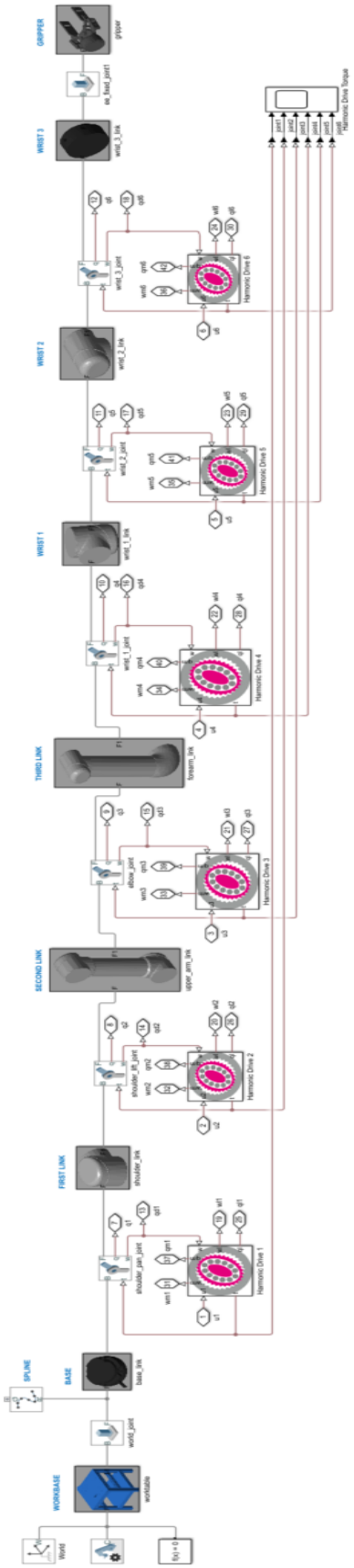


Figure 3.2: UR5 + Harmonic Drive Simscape model

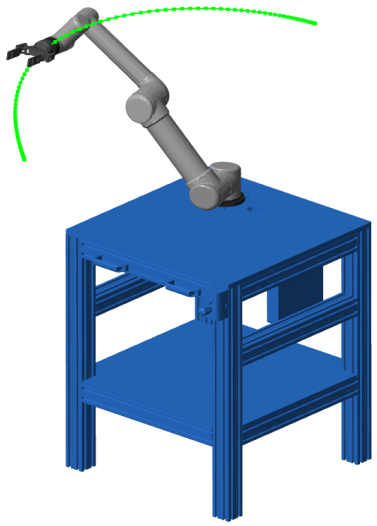


Figure 3.3: UR5 3D model

Harmonic Drive Torque

The following are the Harmonic Drive Torques (t output of each HD), given as input to the corresponding revolute joints.

Considering as final joint configuration $\mathbf{q}_n = [\pi/2 \quad -\pi/2 \quad 0 \quad 0 \quad 0 \quad \pi/2]$:

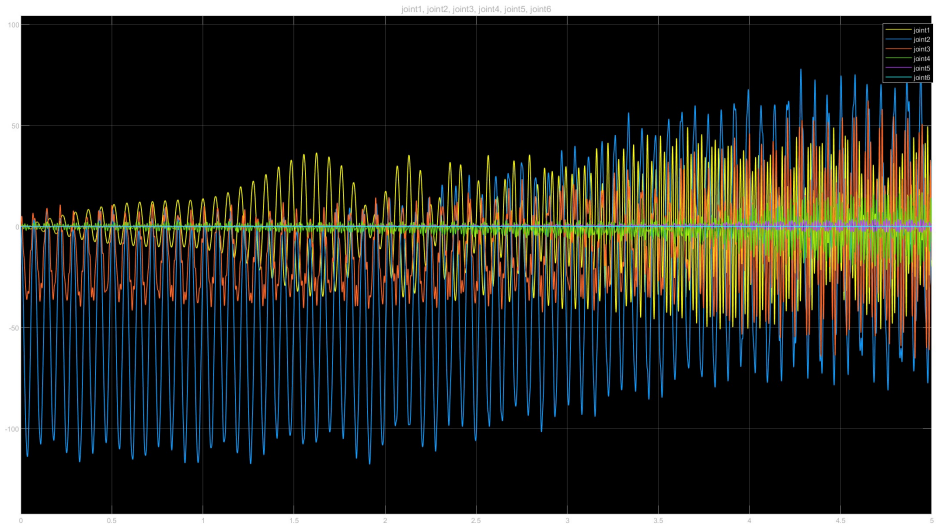


Figure 3.4: Harmonic drive Torque when $\mathbf{q}_n = [\pi/2 \quad -\pi/2 \quad 0 \quad 0 \quad 0 \quad \pi/2]$

If final joint configuration is $\mathbf{q}_n = [\pi/4 \quad -3\pi/4 \quad \pi/6 \quad 0 \quad \pi/6 \quad \pi/2]$:

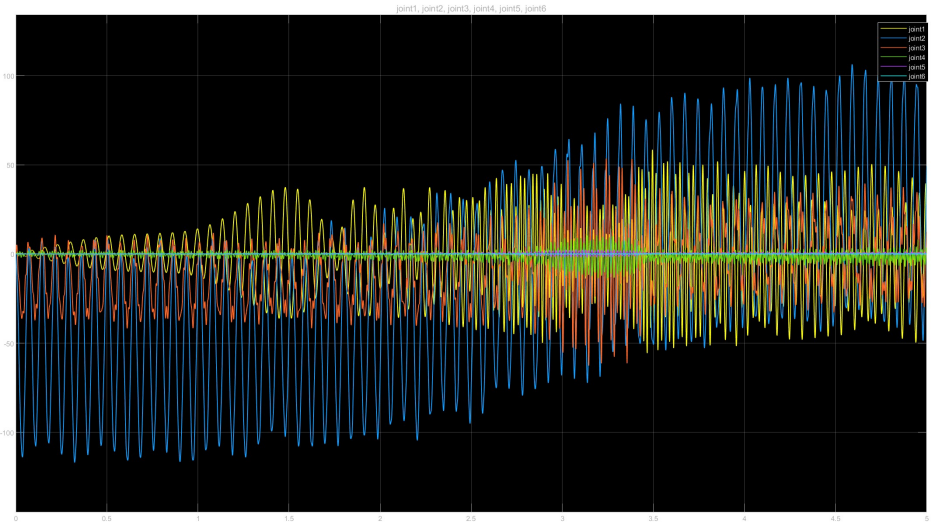
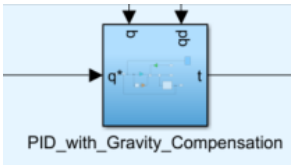


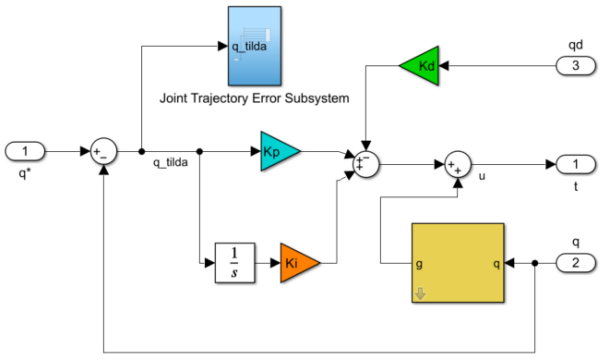
Figure 3.5: Harmonic drive Torque when the final configuration is $\mathbf{q}_n = [\pi/4 \quad -3\pi/4 \quad \pi/6 \quad 0 \quad \pi/6 \quad \pi/2]$

3.1.2 PID with Gravity Compensation

This block, shown in Fig.3.6, considers both the dynamics of movement and the effect of gravity, in order to ensure precise and controlled movement. It is used to control the UR5 manipulator to follow the desired trajectory. In particular, it calculates the torque necessary to make the robot move according to the desired trajectory, counteracting the effects of gravity on the robot.



(a) PID with gravity compensation block



(b) PID with gravity compensation scheme

Figure 3.6: PID with gravity compensation controller.

From the Fig.3.6(a), the block takes the following input:

- $\mathbf{q}^*$: Desired joint position

- **q**: Current joint position
- **qd**: Current joint angular velocity

and returns as output **t**, that represents the control torque to move the UR5 manipulator. The scheme presented in Fig.3.6(b), is obtained considering the theory explained in section 1.5.2. The yellow block is the *gravload* block [1], that allows to compute the joint gravity loading $(1 \times N)$ for the robot UR5 in the current joint configuration $\mathbf{q}\,(1 \times N)$ (so, the gravity joint force), where N is the number of robot joints.

Joint Trajectory Error

Within the PID block with gravity compensation there is the possibility of analyzing the joint trajectory error (Joint Trajectory Error Subsystem). Joint analysis of trajectory errors is important in robotics and automation. In many robotic applications it is essential that the robot performs precise and accurate movements. *The error in the joint trajectory provides crucial information about the discrepancy between the desired trajectory and the one actually followed by the robot.* Such analysis is critical to ensuring optimal performance, safety and reliability in robotic and automated systems. The results are the following: Considering as final joint configuration $\mathbf{q}_n = [\pi/2 \quad -\pi/2 \quad 0 \quad 0 \quad 0 \quad \pi/2]$:

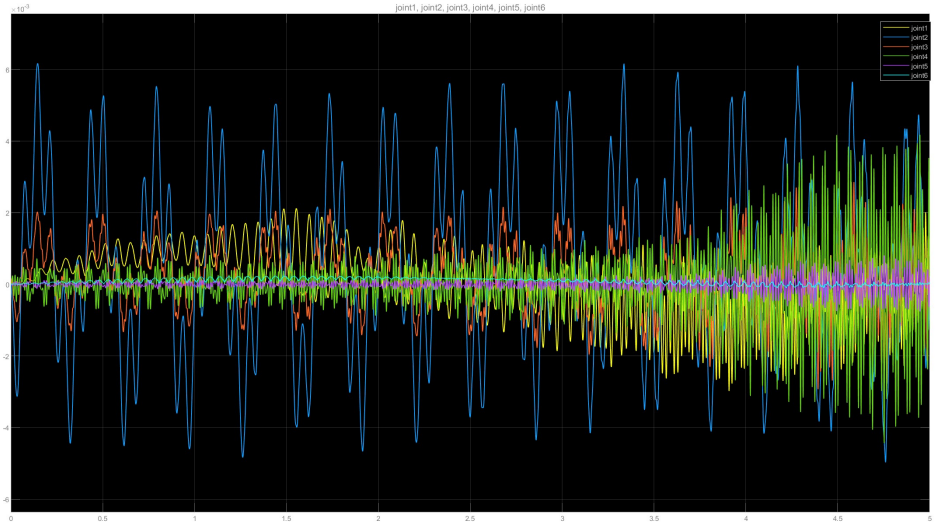


Figure 3.7: Joint Trajectory Error when $\mathbf{q}_n = [\pi/2 \quad -\pi/2 \quad 0 \quad 0 \quad 0 \quad \pi/2]$

If final joint configuration is $\mathbf{q}_n = [\pi/4 \quad -3\pi/4 \quad \pi/6 \quad 0 \quad \pi/6 \quad \pi/2]$:

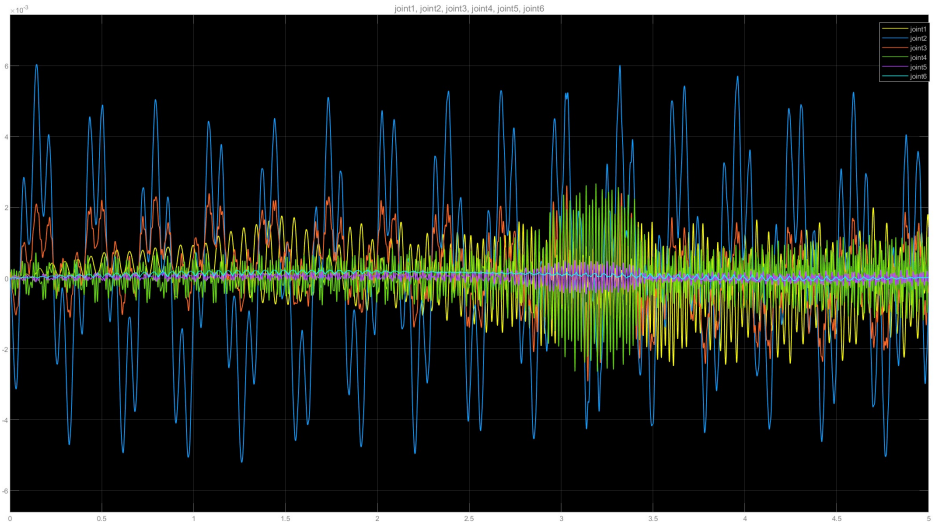


Figure 3.8: Joint Trajectory Error when the final configuration is $\mathbf{q}_n = [\pi/4 \quad -3\pi/4 \quad \pi/6 \quad 0 \quad \pi/6 \quad \pi/2]$

As we can see from Fig.3.7 and Fig.3.8 the error is different from 0, but it is very small in both cases and for some applications it is acceptable. Each error is of the order of 10^{-3}.

3.1.3 Joint space and Workspace trajectory subsystem Block

The joint space and Workspace trajectory Subsystem block 3.9 is used only to visualize the trajectories. The block takes only two inputs $\mathbf{q}$ and $\mathbf{q}^*$ Fig.3.10. Where $\mathbf{q}$ represent the current angular position of the joints (it is the output of the Harmonic Drive q_l) and $\mathbf{q}^*$ the desired one.

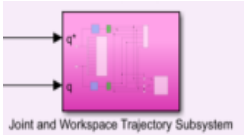


Figure 3.9: Joint and Workspace Trajectory subsystem Block

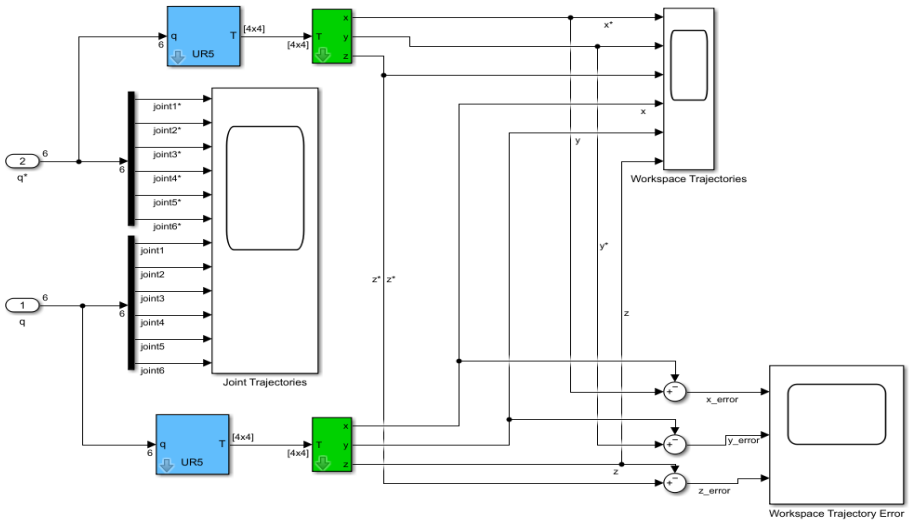


Figure 3.10: Joint and Workspace Trajectory subsystem

As can be seen from the figure are used *fkine* and *T2xyz*. These blocks are taken from the Robotics Toolbox for Matlab (release 10) provided by Peter Corke [1]. They implements the following function:

- fkine(): it is used to calculate the direct kinematics of a robot. This means that it takes as input the robot's joint configurations and calculates the position and orientation of the robot's end-effector in space. The output of the function is the homogeneous transformation matrix that describes the position and orientation of the End-effector with respect to a defined reference system

- T2xyz(): this function is used to extract the position (coordinates) of the End-Effector or any point of interest from the homogeneous trasformation matrix T representing the pose (Position and orientation) of a robot's End-Effector in 3D space.

Joint Space trajectories

Joint spatial trajectories refer to the path followed by the joints of a robotic manipulator over time. In robotics and control theory, it is crucial to plan and control the motion of robotic joints to achieve desired tasks effectively and efficiently. In this case the joint spatial trajectories are generated using the interpolation technique (as explained in the 1.2.2 section, we use a polynomial interpolation method to generate trajectories between two points in the joint space). The results of the **Joint Trajectories Scope** are as follows:

Considering as final joint configuration $\mathbf{q}_n = [\pi/2 \quad -\pi/2 \quad 0 \quad 0 \quad 0 \quad \pi/2]$:

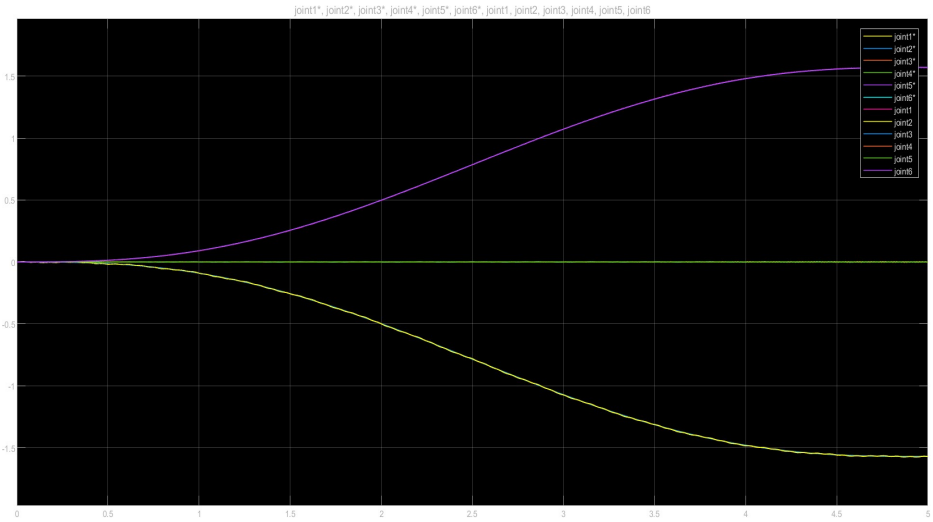


Figure 3.11: Joint Space Trajectories when $\mathbf{q}_n = [\pi/2 \quad -\pi/2 \quad 0 \quad 0 \quad 0 \quad \pi/2]$

If final joint configuration is $\mathbf{q}_n = [\pi/4 \quad -3\pi/4 \quad \pi/6 \quad 0 \quad \pi/6 \quad \pi/2]$:

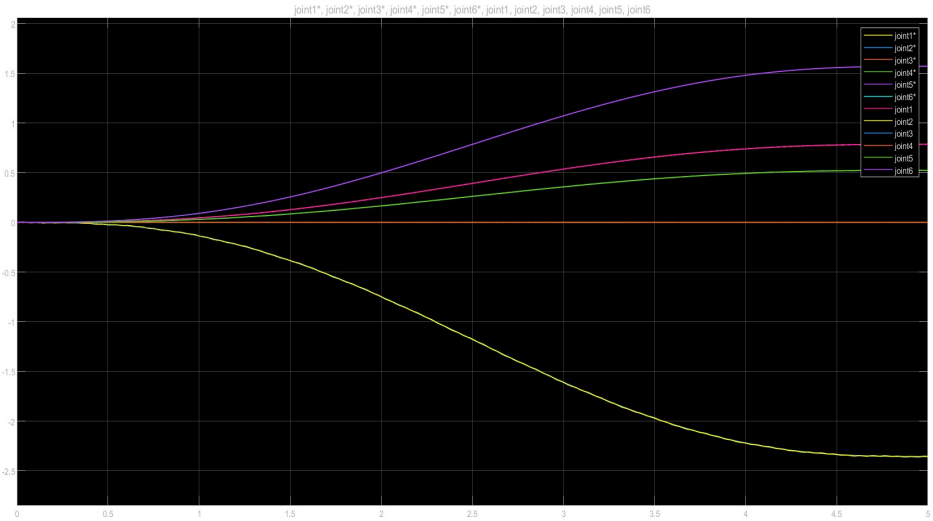


Figure 3.12: Joint Space Trajectories when the final configuration is $\mathbf{q}_n = [\pi/4 \quad -3\pi/4 \quad \pi/6 \quad 0 \quad \pi/6 \quad \pi/2]$

As we can see from Fig.3.11 and Fig.3.12, in both cases, there is a perfect tracking of the desired joint trajectories (indicated with $\star$).

Workspace trajectories

In robotics, workspace trajectories refer to the path followed by a robot's End Effector (tool or manipulator) as it moves within its workspace. In this case the trajectories are defined starting from the joint spatial trajectories using the direct (forward) dynamics algorithm, as explained in the 1.2.2 section. The results of the **Workspace Trajectories Scope** are as follows:

Considering as final joint configuration $\mathbf{q}_n = [\pi/2 \quad -\pi/2 \quad 0 \quad 0 \quad 0 \quad \pi/2]$:

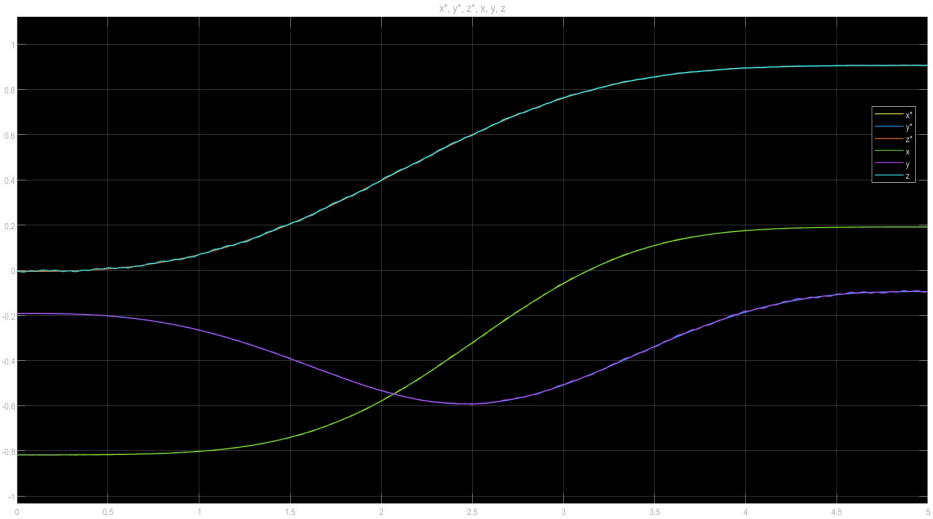


Figure 3.13: Workspace Space Trajectories when $\mathbf{q}_n = [\pi/2 \quad -\pi/2 \quad 0 \quad 0 \quad 0 \quad \pi/2]$

If final joint configuration is $\mathbf{q}_n = [\pi/4 \quad -3\pi/4 \quad \pi/6 \quad 0 \quad \pi/6 \quad \pi/2]$:

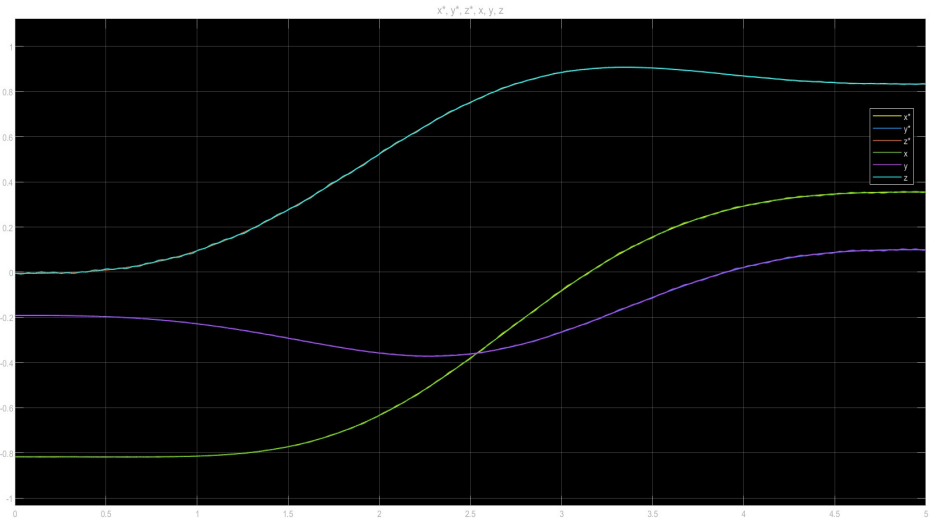


Figure 3.14: Workspace Space Trajectories when the final configuration is $\mathbf{q}_n = [\pi/4 \quad -3\pi/4 \quad \pi/6 \quad 0 \quad \pi/6 \quad \pi/2]$

As we can see from Fig.3.13 and Fig.3.14, in both cases, there is a perfect tracking of the desired Workspace trajectories (indicated with $\star$).

Workspace trajectories Error

In the Joint and Workspace Trajectory subsystem block there is also the possibility of viewing the error of the workspace trajectories. Minimizing these errors is critical to achieving the desired results in terms of performance, safety, efficiency and quality. The results of the **Workspace Trajectory Error Scope** are the following:

Considering as final joint configuration $\mathbf{q}_n = [\pi/2 \quad -\pi/2 \quad 0 \quad 0 \quad 0 \quad \pi/2]$:

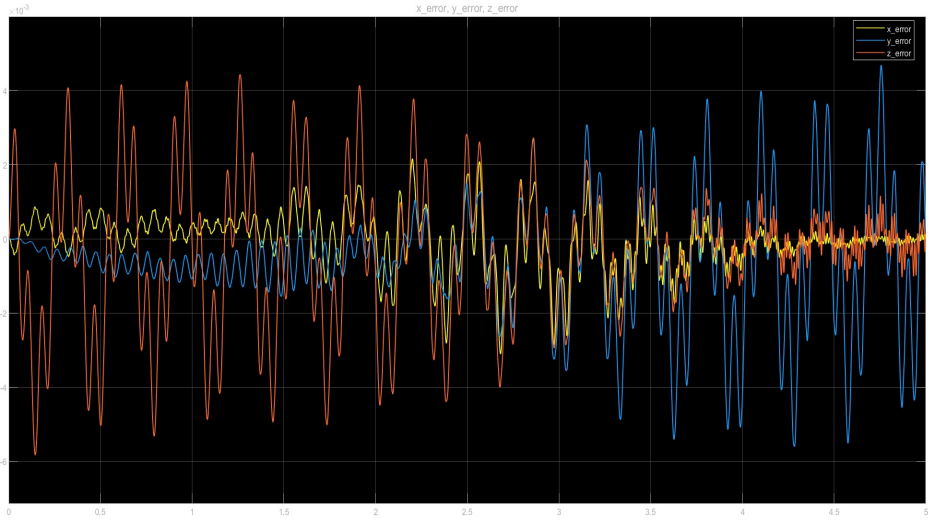


Figure 3.15: Workspace joint trajectory errors when $\mathbf{q}_n = [\pi/2 \quad -\pi/2 \quad 0 \quad 0 \quad 0 \quad \pi/2]$

If final joint configuration is $\mathbf{q}_n = [\pi/4 \quad -3\pi/4 \quad \pi/6 \quad 0 \quad \pi/6 \quad \pi/2]$:

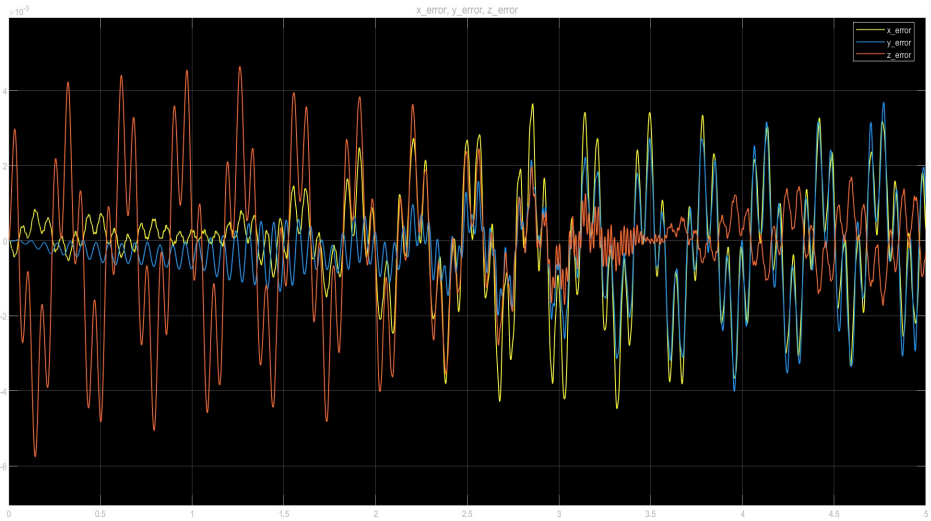


Figure 3.16: Workspace joint trajectory errors when the final configuration is $\mathbf{q}_n = [\pi/4 \quad -3\pi/4 \quad \pi/6 \quad 0 \quad \pi/6 \quad \pi/2]$

As we can see from Fig.3.15 and Fig.3.16, in both cases, errors are different to zero, but are very small. Each error is of the order of 10^{-3}.

Conclusions

In this project, we successfully implemented the UR5, a 6-DOF robotic manipulator featuring a harmonic drive system. The initial phase involved implementing the robotic manipulator and calculating its forward kinematics. Subsequently, we applied PID control method to achieve accurate trajectory tracking. A comprehensive study and modeling of the harmonic drive system were conducted using Simscape based on its bond graph. To enhance control, we incorporated an LQR controller, accounting for the complexities introduced by the harmonic drive and the elasticity within the robot manipulator. In conclusion, the simulation results, including joint-space and workspace trajectories, as well as tracking errors, were effectively presented and analyzed.

Bibliography

[1] Peter I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in MATLAB*. Springer, second edition, 2017. ISBN 978-3-319-54413-7.

[2] Nenad M. Kircanski and Andrew A. Goldenberg. An experimental study of nonlinear stiffness, hysteresis, and friction effects in robot joints with harmonic drives and torque sensors. *The International Journal of Robotics Research*, 16(2):214–239, 1997.

[3] Katharina Kufieta. Force estimation in robotic manipulators: Modeling, simulation and experiments. *Department of Engineering Cybernetics NTNU Norwegian University of Science and Technology*, 2014.

[4] Elias Saerens, Stein Crispel, Pablo Lopez Garcia, Tom Verstraten, Vincent Ducastel, Bram Vanderborght, and Dirk Lefeber. Scaling laws for robotic transmissions. *Mechanism and Machine Theory*, 140:601–621, 10 2019.

[5] Zhiguo Shi, Yuankai Li, and Guangjun Liu. Adaptive torque estimation of robot joint with harmonic drive transmission. *Mechanical Systems and Signal Processing*, 96:1–15, 2017.

[6] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[7] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2005.

[8] Hamid Taghirad. A nonlinear model for harmonic drive friction and compliance. 08 1997.