UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Mechatronics Systems Modelling and Control M

Implementation of a Model of the UR-5 Robotic Arm and Harmonic Drive Actuation System

Professor: **Alessandro Macchelli**

Students:
Burhan Mohayaddin
Ahmed Ahmed

Academic year 2020/2021

Abstract

The role of robots in industrial applications is undeniable as they aid in increasing the throughput and efficiency. Among these robots, robotic manipulators gather special attention. Due to the aforementioned reason, in this work we propose 3D model of UR5 robotic manipulator which has 6 degree of freedom. Moreover, the robot manipulator is not considered as rigid, but with flexible joints which include harmonic drive system. Finally, controller is constructed to control the robot manipulator with harmonic drive system. The whole system is applied in Matlab/Simscape to prove the results.

Contents

List of Figures

Introduction

In this project the modelling of an anthropomorphic 6-DOF robotic arm is implemented. In order to achieve this, Homogenous Transformation matrices are calculated for each link of the robot manipulator according to the predefined 'DH' DenavitâHartenberg parameters. The overall model is implemented using Matlab Simscape Multibody and Simulink tools. To control the robot manipulator, PID + gravity compensation controller is implemented. Moreover, Harmonic Drive is modelled and implemented for each joint. So, additionally LQR controller is added to take into account the dynamics of Harmonic Drive. Finally, joint-space trapezoidal trajectory is generated for tracking.

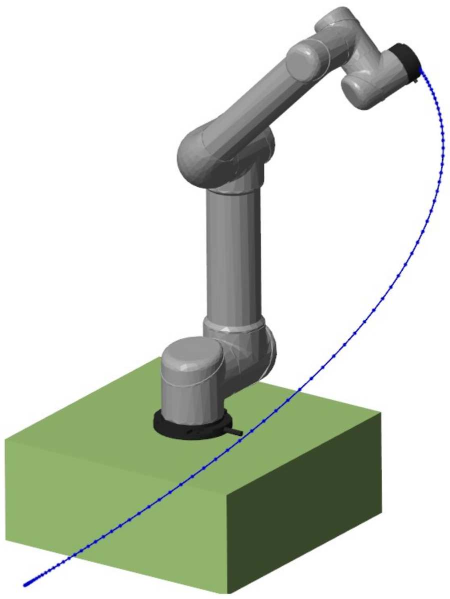


Figure 1: An example graph

Chapter 1

UR5 Robot Arm Simulation

1.1 Forward Kinematics

Before continuing on rest of the simulation of robot manipulator, it is required to obtain forward kinematics of UR5. Robot arms, or serial-link manipulators consist of links and joints that provide motion to the links.

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.

$$R_{n-1}^{n} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{1.1}$$

$$d_{n-1}^{n} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \tag{1.2}$$

$$H_{n-1}^{n} = \begin{bmatrix} R_{n-1}^{n} & d_{n-1}^{n} \\ 0 & 1 \end{bmatrix} \tag{1.3}$$

```matlab
function [H] = calculateForwardKinematics(robot, q)

a   = robot.a;
d   = robot.d;
alp = robot.alp;

H = sym(zeros(4, 4, 6));

for i = 1:6
```

```
    H(:,:,i) = [cos(q(i)) -sin(q(i))*cos(alp(i)) sin(q(i))*sin(
    alp(i)) a(i)*cos(q(i));
                sin(q(i)) cos(q(i))*cos(alp(i)) -cos(q(i))*sin(
    alp(i)) a(i)*sin(q(i));
                0                 sin(alp(i))             cos(alp
    (i))        d(i)       ;
                0                      0                        0
                  1         ];
end

% H6_0 = H(:,:,1) * H(:,:,2) * H(:,:,3) * H(:,:,4) * H(:,:,5) *
     H(:,:,6);
end
```

1.2 PID + Gravity Compensation controller

Generally, the task of the robot arm includes tracking the input trajectory as smooth and precise as possible. So, it is desired to find the structure of the controller which ensures global asymptotic stability of the posture.

Desired control input that stabilizes the system around the equilibrium posture can be obtained by Lyapunov direct method.

The state of the system is given by below vector:

$$x = \left[\widetilde{q}^T \quad \dot{q}^T\right]^T \tag{1.4}$$

$$\widetilde{q} = q_d - q \tag{1.5}$$

The candidate Lyapunov function can be choosen as:

$$V(\dot{q}, \widetilde{q}) = \frac{1}{2}\dot{q}^T B(q)\dot{q} + \frac{1}{2}\widetilde{q}^T K_p \widetilde{q} > 0 \qquad \forall \dot{q}, \widetilde{q} \neq 0 \tag{1.6}$$

where K_p is (nxn) symmetric positive definite matrix.

In order to obtain stability, the derivative of Lyapunov function has to be made nonpositive. So, firstly derivative of the Lyapunov function has to be obtained which is given below:

$$\dot{V}(\dot{q}, \widetilde{q}) = \frac{1}{2}\dot{q}^T(\dot{B}(q) - 2C(q, \dot{q}))\dot{q} - \dot{q}^T F \dot{q} + \dot{q}^T(u - g_{(q)} - K_p \widetilde{q}) \tag{1.7}$$

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

$$N = \dot{B} - 2C \tag{1.8}$$

The second term is negative definite. So, input has to be chosen as:

$$u = g_{(q)} + K_p \widetilde{q} \tag{1.9}$$

To get negative semi-definite $\dot{V}$. Additional damping term K_d can be added to the control law as given below:

$$u = g_{(q)} + K_p \widetilde{q} - K_d \dot{q} \tag{1.10}$$

to improve system time response, where K_d is positive definite matrix.

Overall the control law corresponds to nonlinear compensation action of gravitational terms with a linear proportional-derivative action.

1.2.1 Gravity Compensation

In order to realize PD and gravity compensation controller, gravity compensator has to be implemented. 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:

$$P_i = m_i g^T r_{ci} \tag{1.11}$$

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):

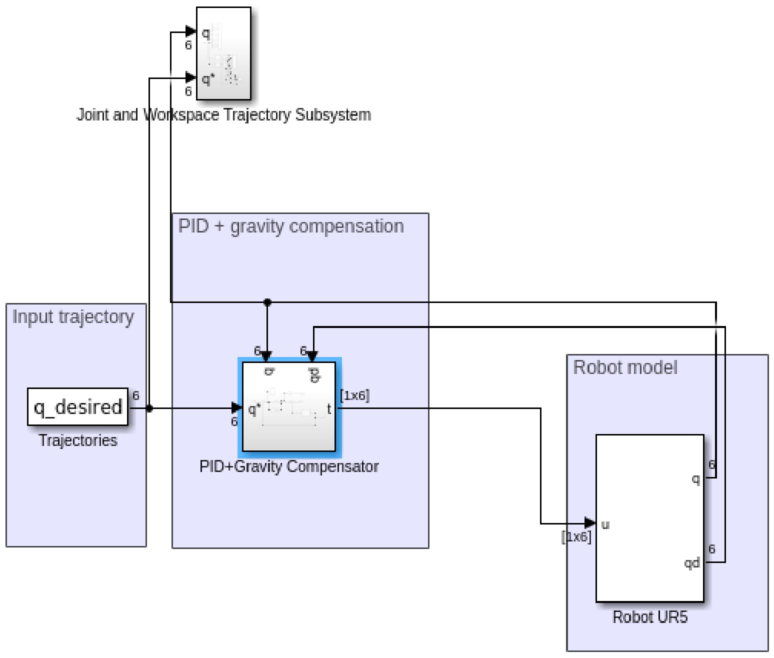
$$P = \sum_{i=1}^{n} P_i = \sum_{i=1}^{n} m_i g^T r_{ci} \tag{1.12}$$

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:

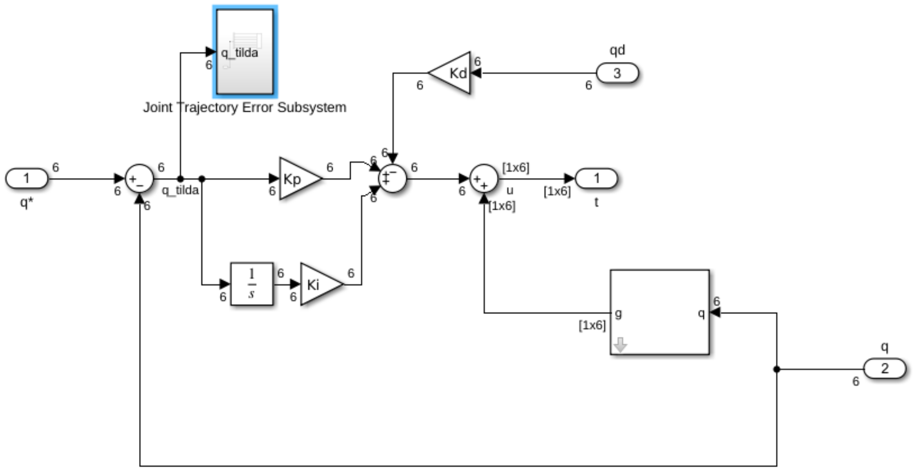
$$g_k = \frac{\partial P}{\partial q_k} \tag{1.13}$$

Overall procedure can be formulated in MATLAB as the following:

```matlab
function [G] = gravityCompensation(robot,q,g_v)

%% Obtain robot parameters
m = robot.m;
center_of_mass = robot.center_of_mass;

%% Calculate forward kinematics
[H] = calculateForwardKinematics(robot, q);

%% Calculate Mass matrix
Hi_0 = sym(zeros(4, 4, 6));
Hi_0(:,:,1) = H(:, :, 1);

pl_i = sym(zeros(3,1,6));
temp = Hi_0(:, :, 1) * center_of_mass(1,:)';
pl_i(:,:,1) = temp(1:3);
```



(a) PD+Gravity compensation controller



(b) Inside the controller

Figure 1.1: (a) PD+Gravity compensation controller and (b) design of the controller

```
for i=2:6
    Hi_0(:, :, i) = Hi_0(:, :, i-1) * H(:, :, i);
    Hi_0(:, :, i) = simplify(Hi_0(:, :, i));
    temp = Hi_0(:, :, i) * center_of_mass(i,:)';
    pl_i(:,:,i) = temp(1:3);
    pl_i(:,:,i) = simplify(pl_i(:,:,i));
end

%% Calculate potential energy
P = sym(zeros(1,6));

for i=1:6
    P(i) = m(i) * g_v * pl_i(:,:,i);
end
P_t = sum(P);

G = sym(zeros(1,6));
for i=1:6
    G(i) = diff(P_t, q(i));
end

%% Save workspace
%save('DynamicsWorkspaceN.mat', 'G');

end
```

1.3 Simulation Results

Once the robot model and the controller has been implemented, simulation is ready to be started. Firstly, random joint-space trajectory is created. In order to visualize the trajectory in workspace inverse kinematics is used to get 3D end-effector coordinates at each time instant. The trajectory can be seen in figure 1.2.

Desired and actual jointspace and workspace trajectories can be seen in figure 1.3 and figure 1.4 respectively.

From figure 1.3 and figure 1.4 it is obvious that trajectory is being tracked satisfactorily by the robot manipulator. This can be observed additionally by inspecting the tracking error both in jointspace and workspace which are depicted in figure 1.5 and figure 1.6 respectively.

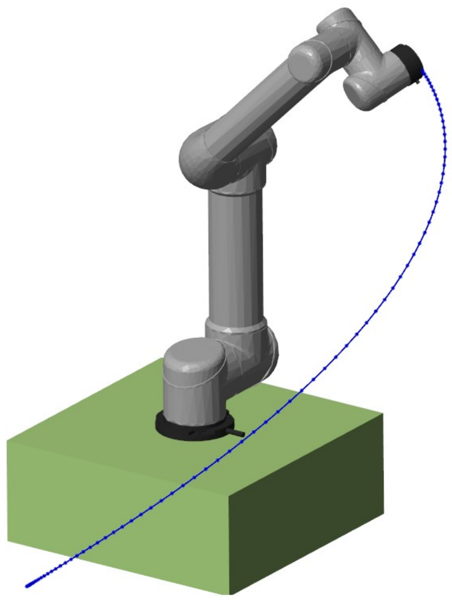


Figure 1.2: Trajectory visualization

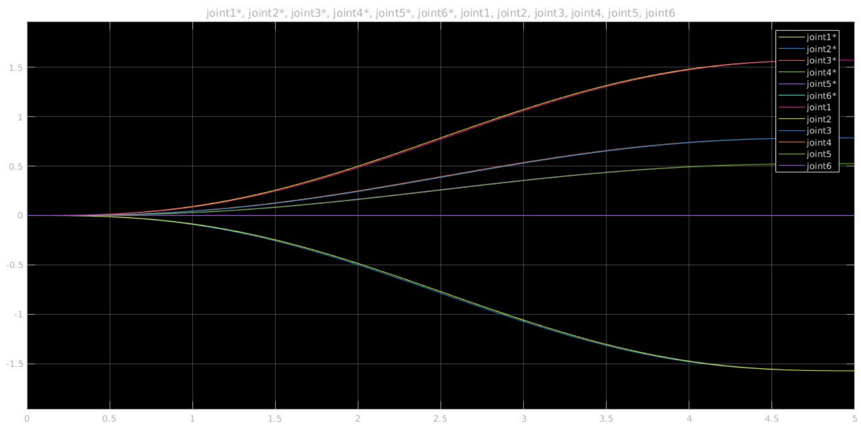


Figure 1.3: Desired and actual jointspace trajectories

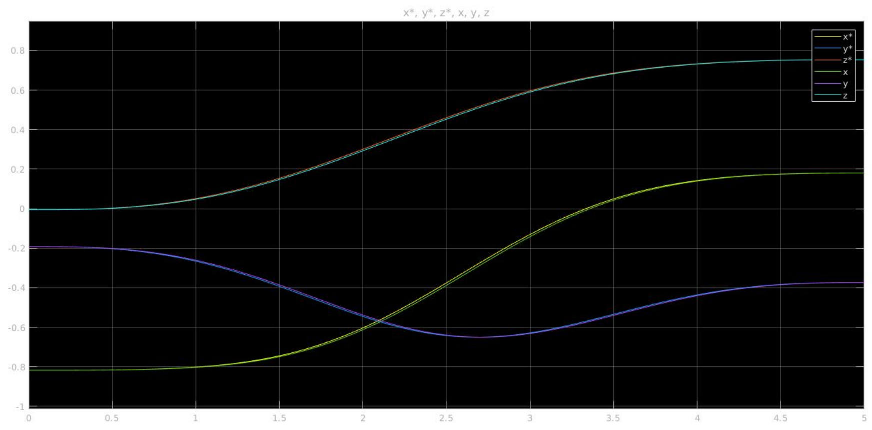


Figure 1.4: Desired and actual workspace trajectories

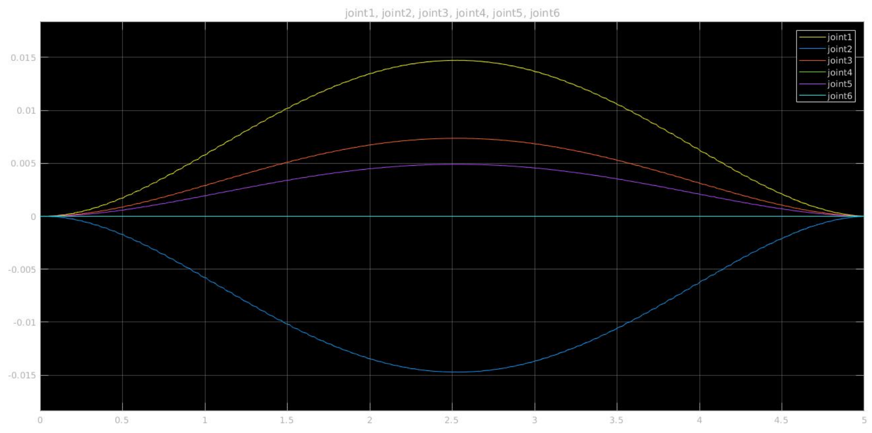


Figure 1.5: Jointspace tracking error

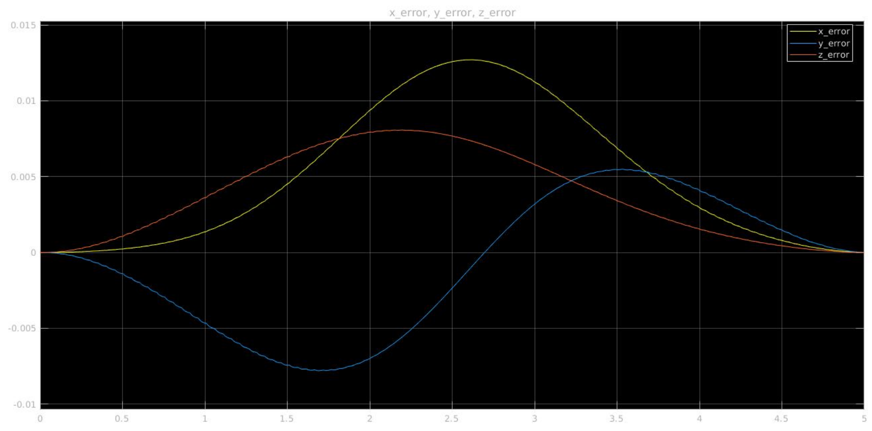


Figure 1.6: Workspace tracking error

Chapter 2

Harmonic Drive Modelling and Simulation

Implementing correct robot drive systems is of paramount importance for achieving smooth low-speed motion and compliant behaviour in contact situations. 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.

Harmonic drive contains these properties more than other drive systems, because of its unconventional gear-tooth meshing action. Namely, harmonic drive mechanism has three basic components:

- wave generator
- flex spline
- circular spline

Wave generator is the input rotation to the system. As the wave generator plug rotates, the flex spline teeth which are meshed with those of the circular spline slowly change position. The major axis of the flex spline's ellipse rotates with wave generator, so the points where the teeth mesh revolve around the center point at the same rate as the wave generator's shaft. The key to the design of the strain wave gear is that there are fewer teeth on the flex spline than there are on the circular spline. This means that for every full rotation of the wave generator, the flex spline would be required to rotate a slight amount (two teeth in this example) backward relative to the circular spline. Thus the rotation action of the wave generator results in a much slower rotation of the flex spline in the opposite direction. So, for the angular velocity of the wave generator we can write:

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

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. For the sake of simplicity, cicular spline will be considered as fixed, so $\dot{q_c}$ will be zero which will simplify the above equation as below:

$$\dot{q_w} = N\dot{q_f} \tag{2.2}$$

2.1 HD Model on Simscape

In order to implement the model of harmonic drive in Simscape, simplified bondgraph of the system is given in figure 2.1:

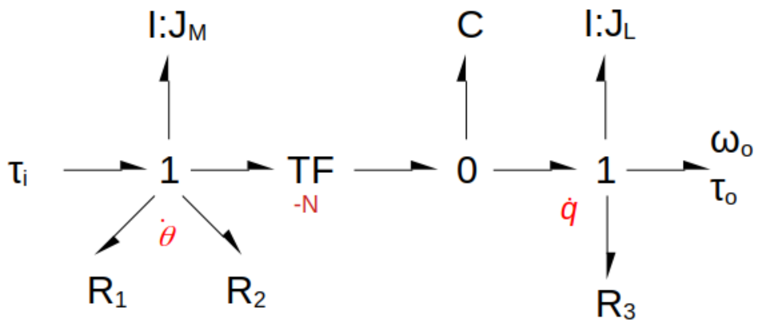


Figure 2.1: Harmonic Drive bond graph

In UR5 robot manipulator two different harmonic drive modules have been utilized, namely for the first three joints HFUS25 and for the last three joints HFUS14 with gear ration of 100.

Based on the bond graph given in figure 2.1, the harmonic drive can be modelled as in figure 2.2.

This model has been created for each joint of the robot manipulator. The input to the subsystem is torque from the controller.

2.1.1 Nonlinear Damper Element

Harmonic drive model contains friction terms due to the internal elements of the harmonic drive. These friction terms are nonlinear and can be modelled by below formula:

$$\tau_b = B\dot{\theta} + (B^+, B^-)sign(\dot{\theta}) \tag{2.3}$$

It is clear from the formula that the friction term changes depending on the sign of the wave generator's velocity.

As these friction terms are nonlinear they have been implemented as custom components in Simscape.

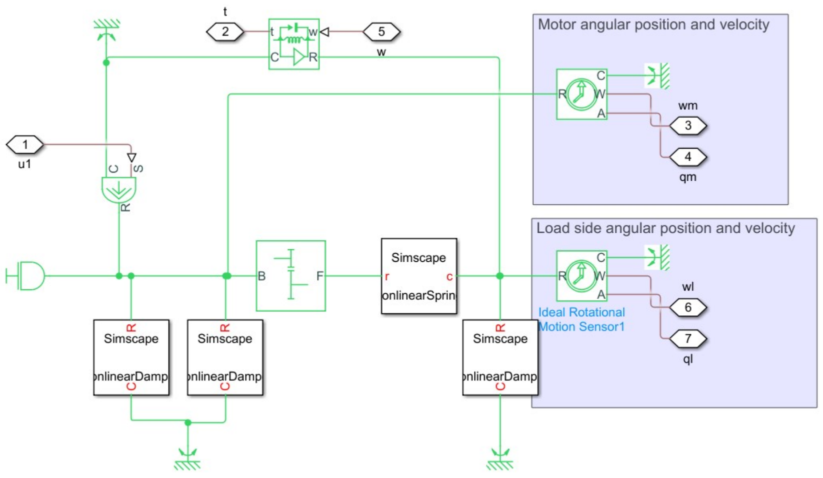

t
2
5
w
1
u1
Motor angular position and velocity
wm
3
4
qm
Load side angular position and velocity
wl
6
7
ql
Ideal Rotational Motion Sensor1
Simscape
onlinearSprin
onlinearDamp
B
F

Figure 2.2: Harmonic drive model

```
component nonlinearDamper

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

parameters
    B = {6.4e-4, 'N*m/(rad/s)'};      % Viscous Coefficient
    B_p = {0.008, 'N*m'};             % Positive-direction Coulumb friction
    B_n = {-0.007, 'N*m'};            % Negative-direction Coulumb friction
end

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

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

equations
    w == r.w - c.w;

    if w > 0
        t == B * w + B_p;
    else
        t == B * w + B_n;
    end
```

```
end

end
```

2.1.2 Nonlinear Spring Element

The key element in harmonic drive model is nonlinear spring element which represents flexspline elasticity. The torsional stiffness can be evaluated based on the below torque-torsion curve. It is obvious from the figure 2.3 that

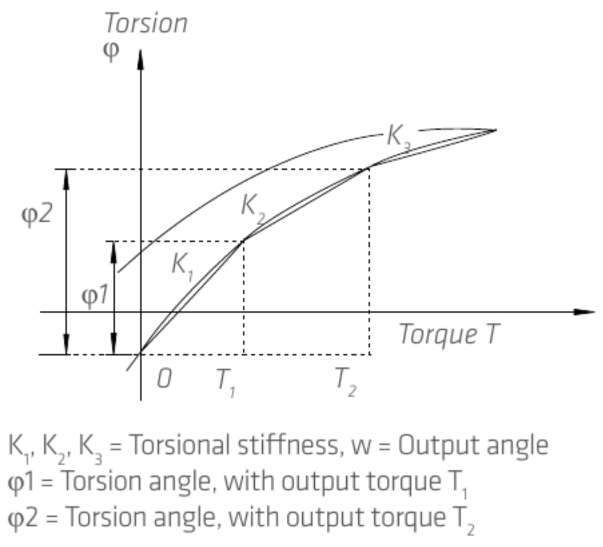


Figure 2.3: Torque-torsion curve

the curve consist of three regions, namely low torque region, middle torque region and high torque region. The torsional stiffness for each region can be calculated by firstly linearizing the overall curve and calculating stiffness values as slope of these curves.

So, torsion angle for each region can be calculated then as:

$$\varphi = \frac{T}{K_1} \qquad (T \leq T_1) \tag{2.4}$$

$$\varphi = \frac{T_1}{K_1} + \frac{T - T_1}{K_2} \qquad (T_1 < T \leq T_2) \tag{2.5}$$

$$\varphi = \frac{T_1}{K_1} + \frac{T_2 - T_1}{K_2} + \frac{T - T_2}{K_3} \qquad (T > T_2) \tag{2.6}$$

However, the input for the spring element is the torsion angle which is obtained by integrating the angular velocity of the flexspline and the output is the torque. So, the formulas have to be reversed. Namely, firstly torsion angles corresponding to torques T_1 and T_2 are calculated which can be called ϕ_1 and ϕ_2 respectively.

$$\varphi_1 = \frac{T_1}{K_1} \tag{2.7}$$

$$\varphi_2 = \frac{T_1}{K_1} + \frac{T_2 - T_1}{K_2} \tag{2.8}$$

Then, the input torsion angle is compared in order to determine the region of activity and so, the corresponding torsional stiffness value. And at the end, output torques are calculated based on determined torsional stiffness values as below:

$$\varphi \leq \varphi_1 \Rightarrow T = \varphi * K_1 \tag{2.9}$$

$$\varphi_1 < \varphi \leq \varphi_2 \Rightarrow T = (\varphi - \varphi_1) * K_2 + T_1 \tag{2.10}$$

$$\varphi > \varphi_2 \Rightarrow T = (\varphi - \varphi_2) * K_3 + T_2 \tag{2.11}$$

In Simscape, custom nonlinear spring element has been implemented as shown below.

```
component nonlinearSpring

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

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

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

equations
```

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

    w == r.w - c.w;
    w == theta.der;

    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
end

end
```

2.2 Controller Design

In the early sections of the report PD and gravity compensation controller has been used to control the robot manipulator while tracking the trajectory. However, after adding the harmonic drive model into the overall robot model, it should also be taken into account in order to track the given trajectory successfully.

In this example, LQR controller will be applied to get the desired motion. However, in order to construct the LQR controller, the differential equations that represent the system has to be obtained. In order to get these equations, bond graph in figure 2.1 has to be assigned with correct causalities. The graph with causalities assigned is given in figure 2.4

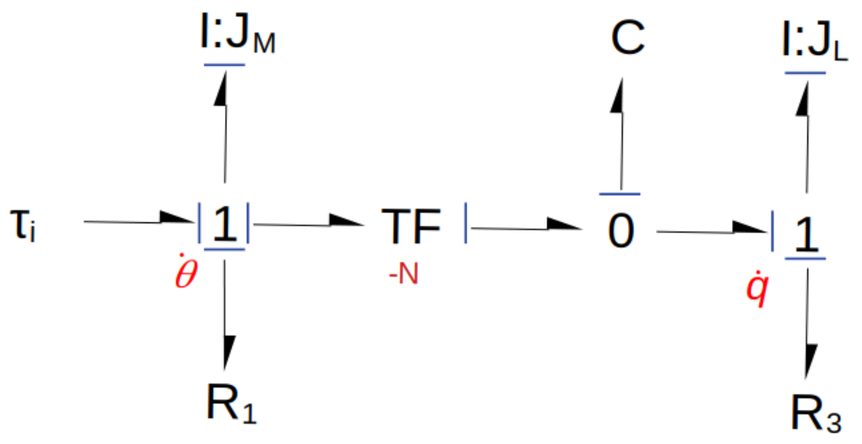


Figure 2.4: Bond graph with causalities

After assigning causalities, it can be noted that there is neither algebraic loop nor derivative causality. So, the state vector is

$$x = \begin{bmatrix} p_m \\ p_l \\ \phi \end{bmatrix} \tag{2.12}$$

After state vector has been determined, bond graph equations can be written in terms of state elements.

$$\dot{p_m} = \tau_i - \frac{K\phi}{N} - \frac{Bp_m}{J_m} \tag{2.13}$$

$$\dot{\phi} = -\frac{p_l}{J_l} + \frac{p_m}{J_m N} \tag{2.14}$$

$$\dot{p_l} = K\phi - \frac{Bp_l}{J_l} \tag{2.15}$$

ϕ can be obtained from equation 2.14 by integrating both sides. If the obtained value of ϕ is plugged into equations 2.13 and 2.15, then below differential equations can be obtained, which represent the system under consideration.

$$J_m\ddot{\theta} + B\dot{\theta} + \frac{K}{N}\left(\frac{\theta}{N} - q\right) = \tau \tag{2.16}$$

$$J_l\ddot{q} + B\dot{q} + K\left(q - \frac{\theta}{N}\right) = 0 \tag{2.17}$$

where:
J_m and J_l: the inertia of motor and load sides
τ_i: the input torque
N: the gear ratio
B: the damping coefficient
K: the spring constant
$\dot{\theta}$: the motor side angular velocity
$\dot{q}$: the load side angular velocity

As 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:

$$\begin{aligned} x_1 &= q \quad x_3 = \theta \\ x_2 &= \dot{q} \quad x_4 = \dot{\theta} \end{aligned} \tag{2.18}$$

So the system equations become:

$$\dot{x_1} = x_2 \tag{2.19}$$

$$\dot{x_2} = -\frac{B}{J_L}x_2 - \frac{K}{J_L}x_1 + \frac{K}{J_L N}x_3 \tag{2.20}$$

$$\dot{x_3} = x_4 \tag{2.21}$$

$$\dot{x_4} = \frac{\tau}{J_m} - \frac{B}{J_m}x_4 - \frac{K}{J_m N^2}x_3 + \frac{K}{J_m N}x_1 \tag{2.22}$$

These equations can be grouped in matrix form as below:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \\ \dot{x_3} \\ \dot{x_4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{K}{J_L} & -\frac{B}{J_L} & \frac{K}{J_L N} & 0 \\ 0 & 0 & 0 & 1 \\ \frac{K}{J_m N} & 0 & -\frac{K}{J_m N^2} & -\frac{B}{J_m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J_m} \end{bmatrix} \begin{bmatrix} \tau \end{bmatrix} \tag{2.23}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{2.24}$$

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:

$$u_{(t)} = -K^T x + u_r \tag{2.25}$$

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

Schematic of the overall controller is illustrated in figure 2.5.

2.3 Simulation Results

After the LQR controller has been implemented and harmonic drive has been modelled, the simulation can be started. Moreover, desired and actual jointspace and workspace trajectories can be seen from figure 2.6 and figure 2.7.

It can be clearly seen from the figures that the given trajectory is tracked successfully both in the jointspace and in the workspace. Furthermore, figure 2.8 and figure 2.9 illustrate that the absolute value of the error is less than $5 * 10^{-3}$ and $4 * 10^{-3}$ in jointspace and workspace respectively.

Moreover, in figure 2.10 unfiltered torque output from harmonic drive is illustrated. From figure 2.10 it is clear that the momentary peak torques for each harmonic drive are within the limit of the device.

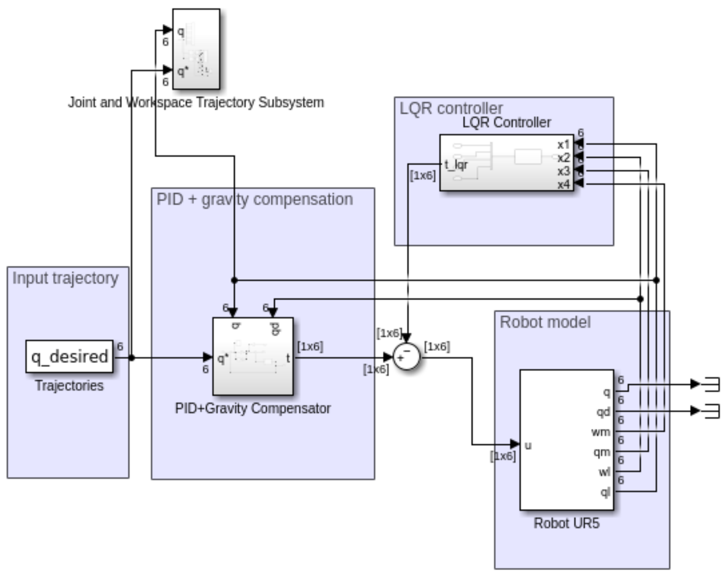


Figure 2.5: PD+gravity compensation and LQR controller

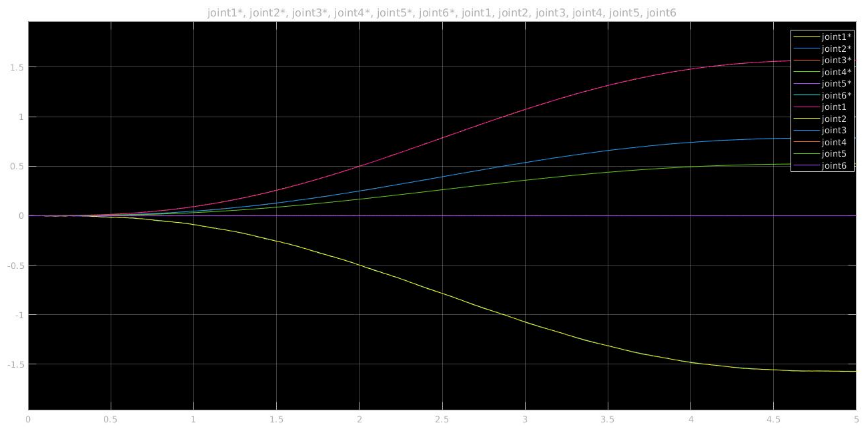


Figure 2.6: Desired and actual jointspace trajectories

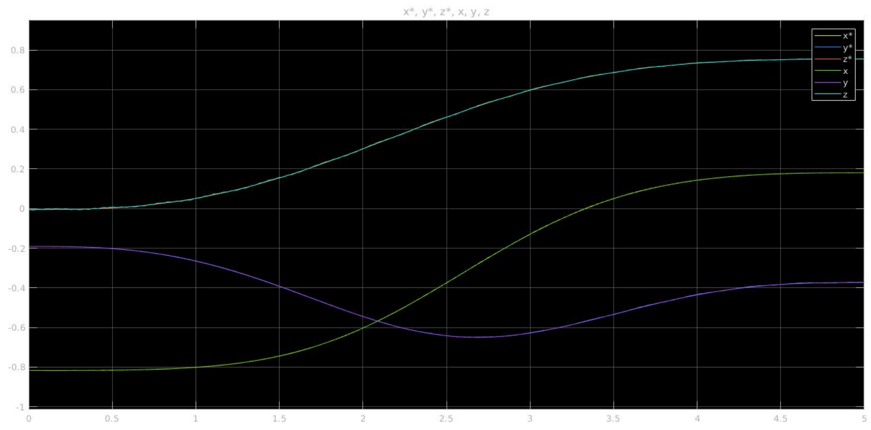


Figure 2.7: Desired and actual workspace trajectories

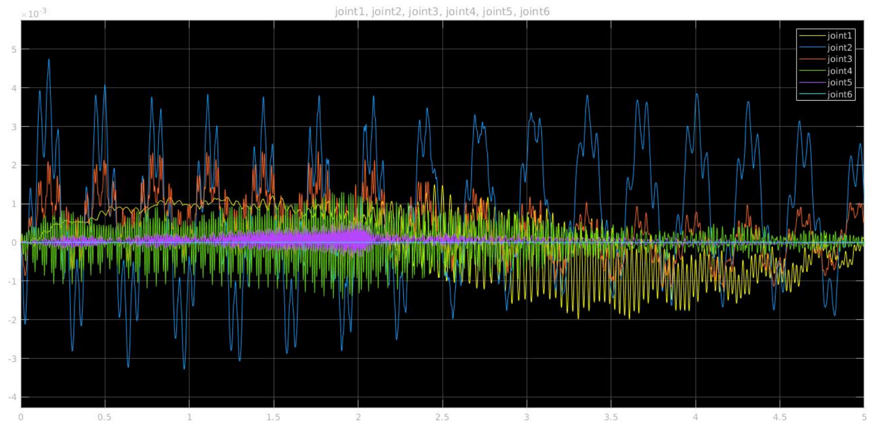


Figure 2.8: Jointspace trajectory error

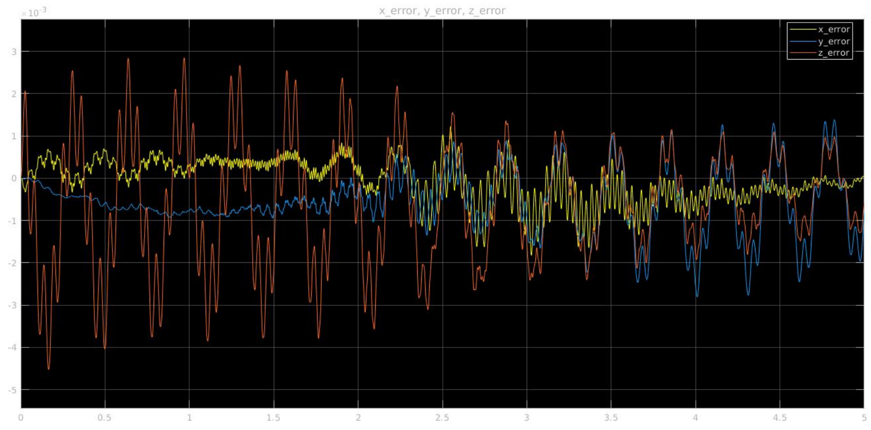


Figure 2.9: Workspace trajectory error

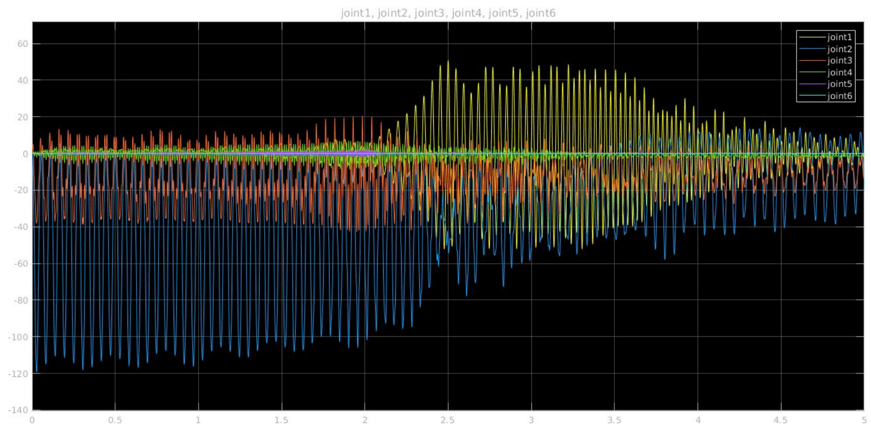


Figure 2.10: Unfiltered torque output from harmonic drives

Conclusions

In this project, UR-5 is a 6-DOF robotic manipulator with harmonic drive system has been implemented. As a first step the robotic manipulator has been implemented and its forward kinematics has been calculated. Secondly, PD and gravity compensation control has been applied in order to track the desired trajectory. Thirdly, harmonic drive system has been studied and modelled in Simscape based on its bond graph. Furthermore, the controller has been improved by implementing additionally LQR controller in order to take into account harmonic drive and so, elasticity in the robot manipulator. Finally, the results of the simulation, namely jointspace and workspace trajectories and tracking errors have been demonstrated and analyzed.

Bibliography

[1] Harmonic Drive AG . Engineering Data HFUS-2UH / 2SO / 2SH Units

[2] MARK W. SPONG, SETH HUTCHINSONM, VIDYASAGAR . Robot Modeling and Control, second edition

[3] Nenad M.Kircanski, Andrew A.Goldenberg . An Experimental Study of Nonlinear Stiffness, Hysteresis, and Friction Effects in Robot Joints with Harmonic Drives and Torque Sensors

[4] Alessandro Macchelli . Kinematic and Dynamic Parameters of the UR5 Robot Manipulator