Project 2
Dynamics and Control

Ganggui Ling gl1920

Kiki yaw sarpong kys254

Temilotzin Monroy xmp205

Part 1

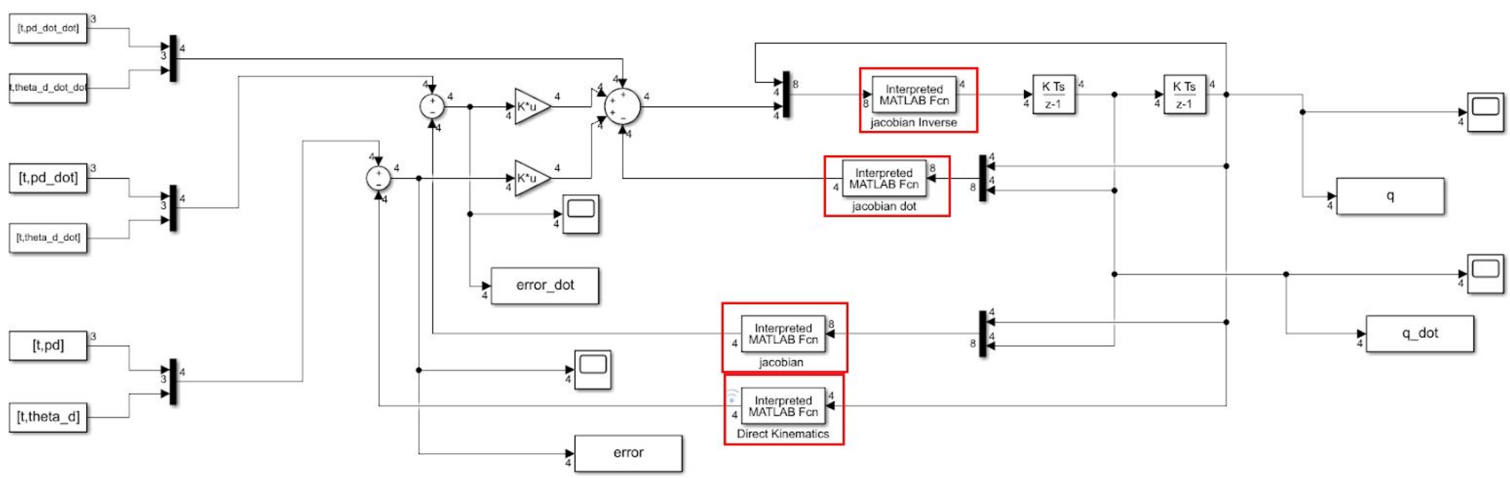


Fig.1 **Second Order Inverse Simulink (New Algorithm**)

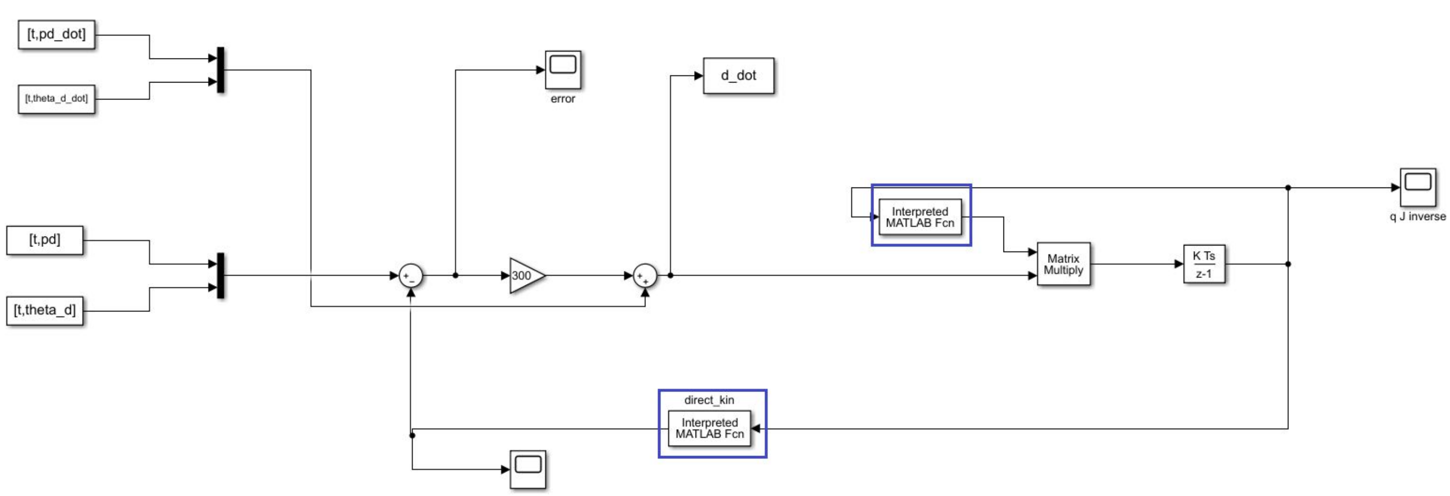


Fig.2 **First Order Inverse Simulink. (Old Algorithm)**

It can be noted that in the implementation of the new algorithm, the Simulink block receives the desired position, velocity and acceleration values in the operational space of the manipulator and converts these values into joint space position, velocity and acceleration. The second order inverse algorithm requires a “Jacobian-dot” function as the derivative of the jacobian of the manipulator, a “jacobian” function from the previous first order system as well the same “jacobian-inverse” and “direct-kinematics” functions. All these functions stay the same as used with the first order system because the second order is just a more advanced first order system.

The difference in both systems can be attributed to the inputs and outputs. The first order system is given the desired position and velocity in the operational space and uses the “jacobian inverse” block and “direct-kinematics” block (shown in blue boxes) to find the joint positions and velocities in the joint space. And the second order system uses the desired end effector position, velocity and acceleration as input to compute the joint position, joint velocity and joint acceleration with “jacobian-inverse”, “jacobian”, “jacobian-dot”, “direct-kinematics” (shown in red boxes).

Jacobian-Dot:

```
[ - (cos(th1 + th2)*(th1_dot + th2_dot))/2 - (th1_dot*cos(th1))/2, -(cos(th1 + th2)*(th1_dot + th2_dot))/2, 0, 0]
[ - (sin(th1 + th2)*(th1_dot + th2_dot))/2 - (th1_dot*sin(th1))/2, -(sin(th1 + th2)*(th1_dot + th2_dot))/2, 0, 0]
[                                                                0,                                        0, 0, 0]
[                                                                0,                                        0, 0, 0]
[                                                                0,                                        0, 0, 0]
[                                                                0,                                        0, 0, 0]
```

Jacobian:

```
[ - sin(th1 + th2)/2 - sin(th1)/2, -sin(th1 + th2)/2,  0, 0]
[   cos(th1 + th2)/2 + cos(th1)/2,  cos(th1 + th2)/2,  0, 0]
[                               0,                 0, -1, 0]
[                               0,                 0,  0, 0]
[                               0,                 0,  0, 0]
[                               1,                 1,  0, 1]
```

Jacobian-Inverse:

[-(2*cos(th1 + th2))/(cos(th1 + th2)*sin(th1) - sin(th1 + th2)*cos(th1)), -(2*sin(th1 + th2))/(cos(th1 + th2)*sin(th1) - sin(th1 + th2)*cos(th1)), 0, 0]

[(2*(cos(th1 + th2) + cos(th1)))/(cos(th1 + th2)*sin(th1) - sin(th1 + th2)*cos(th1)), (2*(sin(th1 + th2) + sin(th1)))/(cos(th1 + th2)*sin(th1) - sin(th1 + th2)*cos(th1)), 0, 0]

[0, 0, -1, 0]

[-(2*cos(th1))/(cos(th1 + th2)*sin(th1) - sin(th1 + th2)*cos(th1)), -(2*sin(th1))/(cos(th1 + th2)*sin(th1) - sin(th1 + th2)*cos(th1)), 0, 1]

Direct-Kinematics:

[cos(th1 + th2)*cos(th4) - sin(th1 + th2)*sin(th4), - cos(th1 + th2)*sin(th4) - sin(th1 + th2)*cos(th4), 0, cos(th1 + th2)/2 + cos(th1)/2]

[cos(th1 + th2)*sin(th4) + sin(th1 + th2)*cos(th4), cos(th1 + th2)*cos(th4) - sin(th1 + th2)*sin(th4), 0, sin(th1 + th2)/2 + sin(th1)/2]

[0, 0, 1, -d3]

[0, 0, 0, 1]

Joint Plots

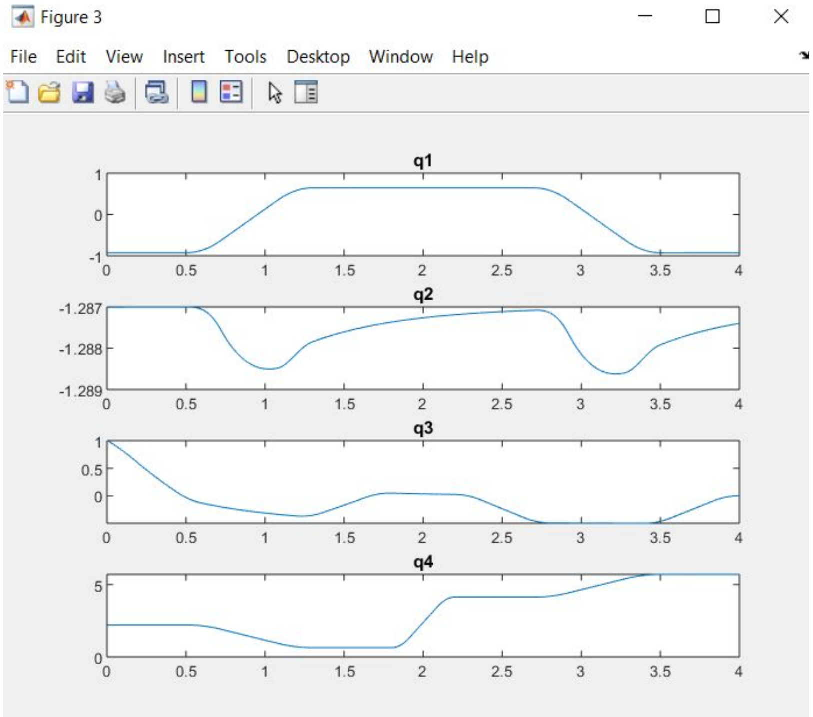


Figure 3
File Edit View Insert Tools Desktop Window Help
q1
q2
q3
q4

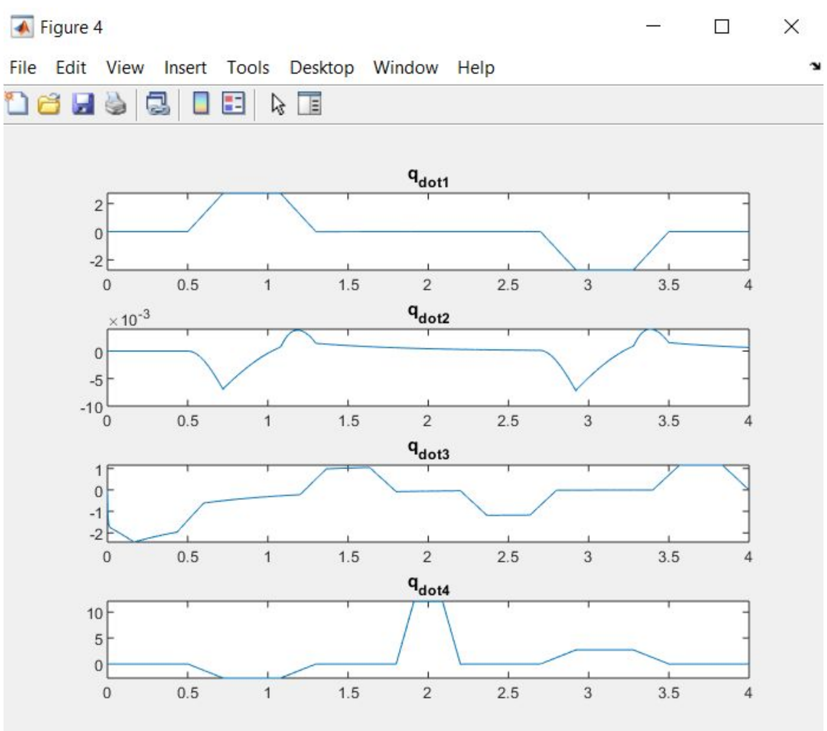


Figure 4
File Edit View Insert Tools Desktop Window Help
q dot1
q dot2
×10^-3
q dot3
q dot4

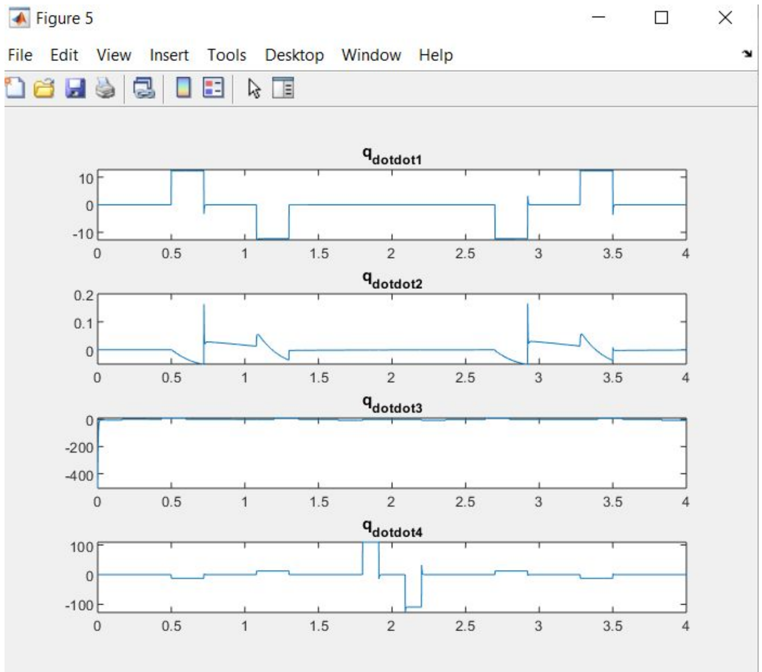


Figure 5
File Edit View Insert Tools Desktop Window Help
$q_{dotdot1}$
$q_{dotdot2}$
$q_{dotdot3}$
$q_{dotdot4}$

Error Plots

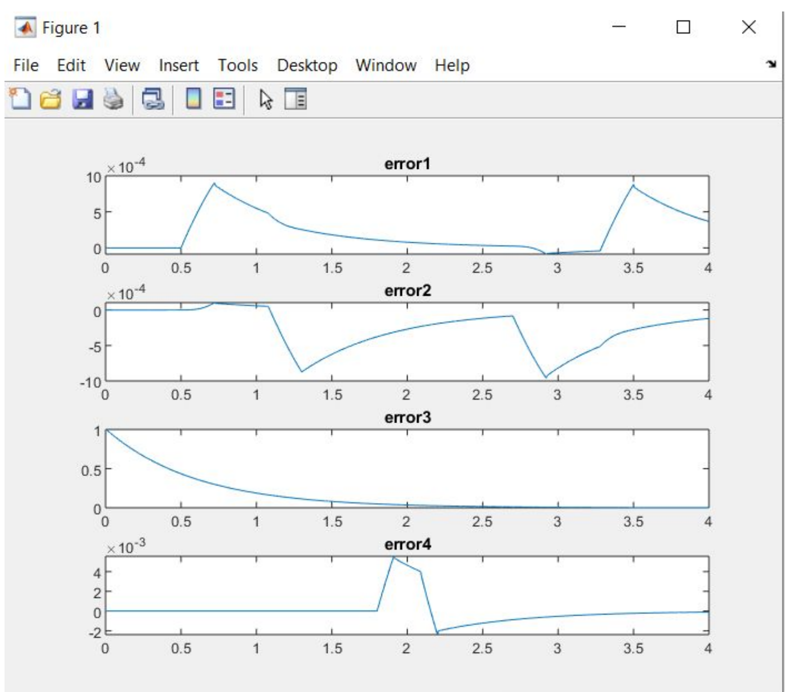
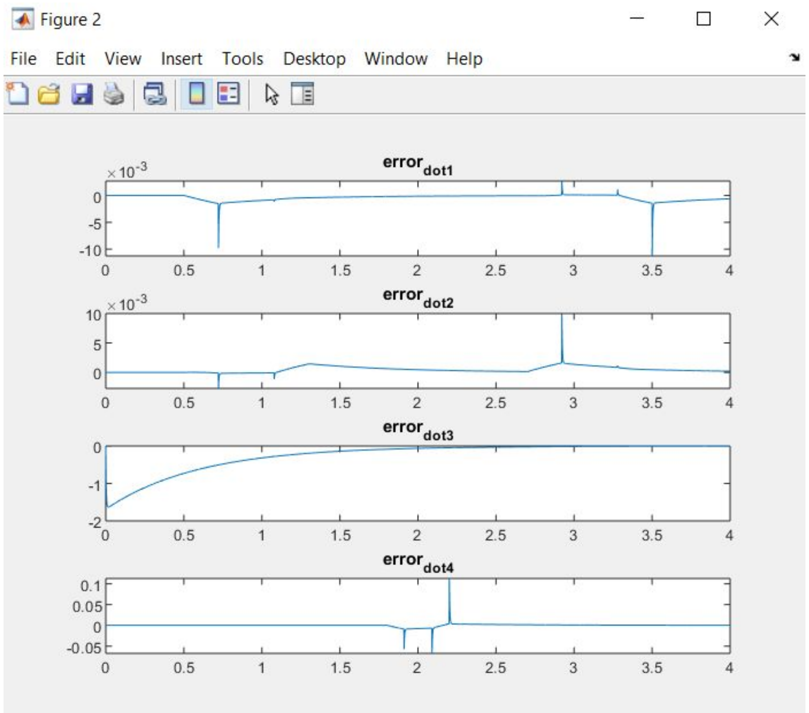


Figure 1
File Edit View Insert Tools Desktop Window Help
error1
error2
error3
error4



Part 2

Dynamic Model

$$\boldsymbol{B}(\boldsymbol{q})\ddot{q}+\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}}+\boldsymbol{F}_v\dot{\boldsymbol{q}}+\boldsymbol{F}_s\operatorname{sgn}(\dot{\boldsymbol{q}})+\boldsymbol{g}(\boldsymbol{q})=\boldsymbol{\tau}-\boldsymbol{J}^T(\boldsymbol{q})\boldsymbol{h}_e$$

The above equation is used to solve the Manipulator Dynamic model. The different variables used are described as:

- **B(q)(q_dotdot):**

$$\boldsymbol{B}(\boldsymbol{q})=\sum_{i=1}^{n}\Big(m_{\ell_i}\boldsymbol{J}_P^{(\ell_i)T}\boldsymbol{J}_P^{(\ell_i)}+\boldsymbol{J}_O^{(\ell_i)T}\boldsymbol{R}_i\boldsymbol{I}_{\ell_i}^i\boldsymbol{R}_i^T\boldsymbol{J}_O^{(\ell_i)}$$

$$+m_{m_i}\boldsymbol{J}_P^{(m_i)T}\boldsymbol{J}_P^{(m_i)}+\boldsymbol{J}_O^{(m_i)T}\boldsymbol{R}_{m_i}\boldsymbol{I}_{m_i}^{m_i}\boldsymbol{R}_{m_i}^T\boldsymbol{J}_O^{(m_i)}\Big)$$

is the $(n\times n)$ *inertia matrix* which is:

- **C(q, q_dot)q_dot:** The term C(q, q˙) accounting for centrifugal and Coriolis forces. The C term is not unique and calculated with the equation

$$
\begin{aligned}
\sum_{j=1}^{n} c_{ij}\dot{q}_j &= \sum_{j=1}^{n}\sum_{k=1}^{n} h_{ijk}\dot{q}_k\dot{q}_j \\
&= \sum_{j=1}^{n}\sum_{k=1}^{n}\left(\frac{\partial b_{ij}}{\partial q_k} - \frac{1}{2}\frac{\partial b_{jk}}{\partial q_i}\right)\dot{q}_k\dot{q}_j.
\end{aligned}
$$

Splitting the first term on the right-hand side by an opportune switch of summation between j and k yields

$$\sum_{j=1}^{n} c_{ij}\dot{q}_j = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial b_{ij}}{\partial q_k}\dot{q}_k\dot{q}_j + \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i}\right)\dot{q}_k\dot{q}_j.$$

As a consequence, the generic element of $\boldsymbol{C}$ is

$$c_{ij} = \sum_{k=1}^{n} c_{ijk}\dot{q}_k \tag{7.44}$$

where the coefficients

$$c_{ijk} = \frac{1}{2}\left(\frac{\partial b_{ij}}{\partial q_k} + \frac{\partial b_{ik}}{\partial q_j} - \frac{\partial b_{jk}}{\partial q_i}\right) \tag{7.45}$$

- **Fv(q_dot):** Viscous friction torques

- **Fs:** Static friction torques. Where Fs is an (n × n) diagonal matrix and sgn (q˙) denotes the (n × 1) vector whose components are given by the sign functions of the single joint velocities.

- **g(q):** Gravity vector

- **T:** Joint actuation torques

- **Jt(q):** Jacobian transpose

- **H_e:** Denotes the vector of force and moment exerted by the end-effector on the environment.

B Function

The B function is the (n x n) inertia matrix calculated by:

$$\boldsymbol{B}(\boldsymbol{q}) = \sum_{i=1}^{n} \Big(m_{\ell_i} \boldsymbol{J}_P^{(\ell_i)T} \boldsymbol{J}_P^{(\ell_i)} + \boldsymbol{J}_O^{(\ell_i)T} \boldsymbol{R}_i \boldsymbol{I}_{\ell_i}^i \boldsymbol{R}_i^T \boldsymbol{J}_O^{(\ell_i)} + m_{m_i} \boldsymbol{J}_P^{(m_i)T} \boldsymbol{J}_P^{(m_i)} + \boldsymbol{J}_O^{(m_i)T} \boldsymbol{R}_{m_i} \boldsymbol{I}_{m_i}^{m_i} \boldsymbol{R}_{m_i}^T \boldsymbol{J}_O^{(m_i)} \Big)$$

is the $(n \times n)$ *inertia matrix* which is:

The calculation for the B function is below:

```
B11 = ml1*l1^2 + Il1 + Im1 * kr1^2 + ml2*(0.25+l2^2+l1*l2) + Il2
      + Im2 + ml3 + Im3 + ml4*(0.5*(1+cos(th2))) + Il4;
B12 = ml2 * (l2^2 + 0.5*l2*cos(th2)) + Il2 + Im2*kr2 + ml3 + Im3
      + ml4*(0.25 * (1 + cos(th2))) + Il4;
B13 = -ml3 - Im3*kr3;
B14 = Il4;
B21 = ml2*(l2^2+0.5*l2*cos(th2)) + Il2 + Im2*kr2 + ml3 + Im3
      + ml4*(0.25*(1+cos(th2))) + Il4;
B22 = ml2*l2^2 + Il2 + Im2*kr2^2 + ml3 + Im3 + ml4*0.25 + Il4;
B23 = -ml3 - Im3*kr3;
B24 = Il4;
B31 = -ml3-Im3*kr3;
B32 = -ml3-Im3*kr3;
B33 = ml3 + Im3*kr3^2;
B34 = 0;
B41 = Il4;
B42 = Il4;
B43 = 0;
B44 = Im4;
OUT = [B11 B12 B13 B14
         B21 B22 B23 B24
         B31 B32 B33 B34
```

```
        B41 B42 B43 B44]
    + Iml4 * [1 1 0 kr4;
              1 1 0 kr4;
              0 0 0 0;
             kr4 kr4 0 kr4^2];
```

n Function

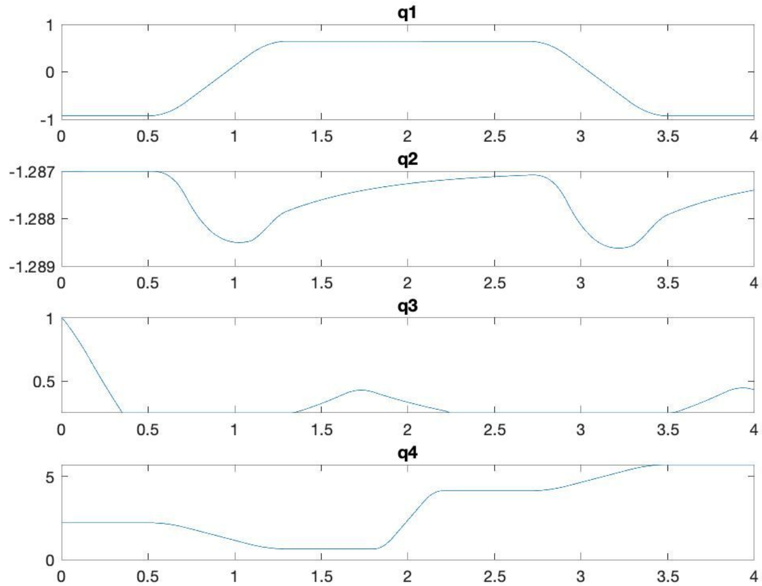
The n-function calculates the vector of Coriolis, centrifugal, gravitational, and damping terms. The equation to calculate n(q,q_dot) is below:

$$n(q,\dot{q}) = C(q,\dot{q})\dot{q} + F\dot{q} + g(q). \qquad (8.56)$$

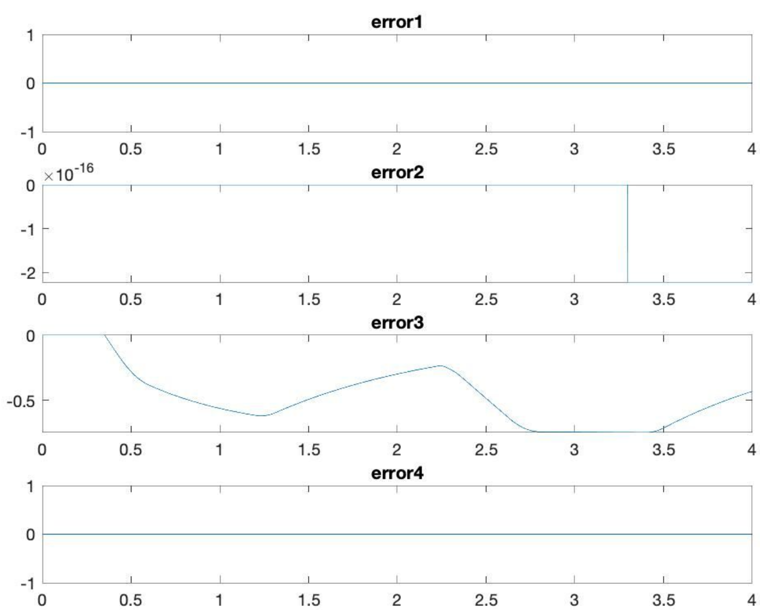
The Calculation of n-function is below:

```
h = -1/2 * sin(th2) * (ml2 * l2 + ml4 * 0.5);

C = [h*th2_dot   2*h*th2_dot 0 0
      -h*th1_dot       0      0 0
      0 0 0 0
      0 0 0 0];

Fv = [0.0001 0 0 0
       0 0.0001 0 0
       0 0 0.01 0
       0 0 0 0.005];

g = [ml3*g ml3*g -ml3g 0];

n = C * joint_velocities + Fv * joint_velocities + g;
```

Joint Plot



Error Plot



Part 3

As perfectly stated in the book, the goal while planning a trajectory is to generate the necessary inputs so that the control system can execute the desired movement in the manipulator.

For this part, we were asked to generate the trajectory for the robot arm to execute determined motions for 4 seconds. We were given a set of precise points where the manipulator should travel without making any sudden stop. This means that, if we analyze the trajectory plot after calculation, there should not be any sharp edges.

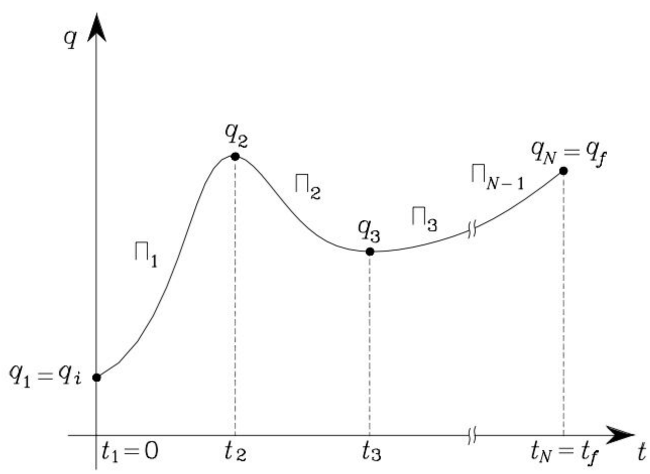


Figure. Planned trajectory.

In the previous figure, we can note how there is no sudden change of direction at any point. This means that the system has some sort of anticipation prior to any change of direction.

In order to achieve this, we need to compute the following equations and variables.

$$s_j(t) = \begin{cases} 0 & 0 \le t \le t_{j-1} - \Delta t_j \\ s'_j(t + \Delta t_j) & t_{j-1} - \Delta t_j < t < t_j - \Delta t_j \\ \|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\| & t_j - \Delta t_j \le t \le t_f - \Delta t_N, \end{cases}$$

$$\boldsymbol{p}_e = \boldsymbol{p}_0 + \sum_{j=1}^{N} \frac{s_j}{\|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|}(\boldsymbol{p}_j - \boldsymbol{p}_{j-1}),$$

$$\dot{\boldsymbol{p}}_e = \sum_{j=1}^{N} \frac{\dot{s}_j}{\|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|}(\boldsymbol{p}_j - \boldsymbol{p}_{j-1}) = \sum_{j=1}^{N} \dot{s}_j \boldsymbol{t}_j$$
$$\ddot{\boldsymbol{p}}_e = \sum_{j=1}^{N} \frac{\ddot{s}_j}{\|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|}(\boldsymbol{p}_j - \boldsymbol{p}_{j-1}) = \sum_{j=1}^{N} \ddot{s}_j \boldsymbol{t}_j,$$

We know that p corresponds to position (which plotted should be the trajectory). If we take out the first derivative (q_dot) we should be able to obtain the velocity. After that, the second derivative (q_dot_dot) should give us the acceleration.

The outputs obtained show us the correct functionality of the system. Which can be found next.

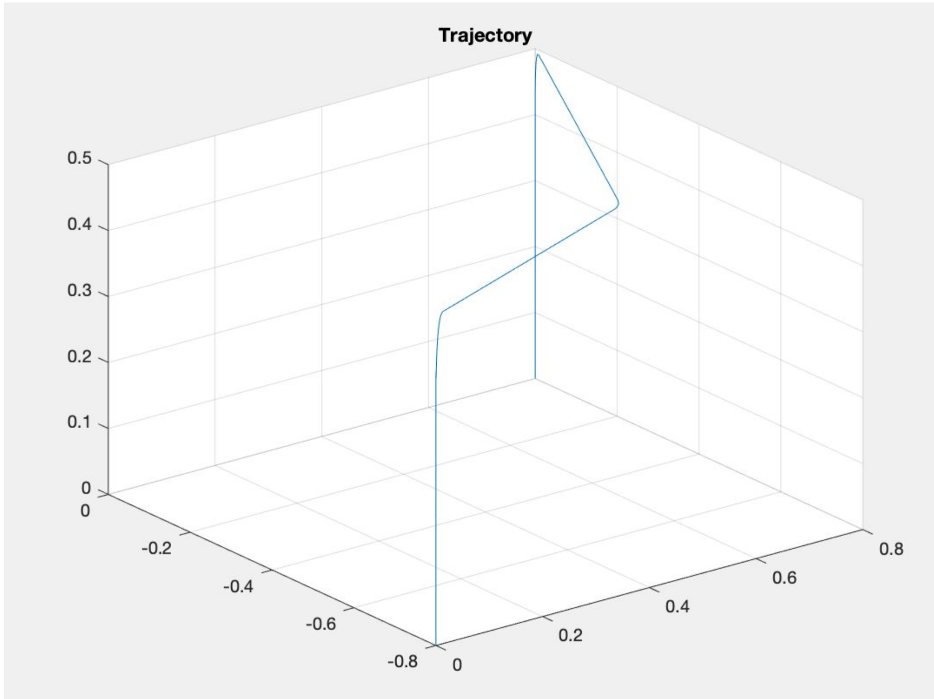


Figure. Plot for trajectory of the end effector.

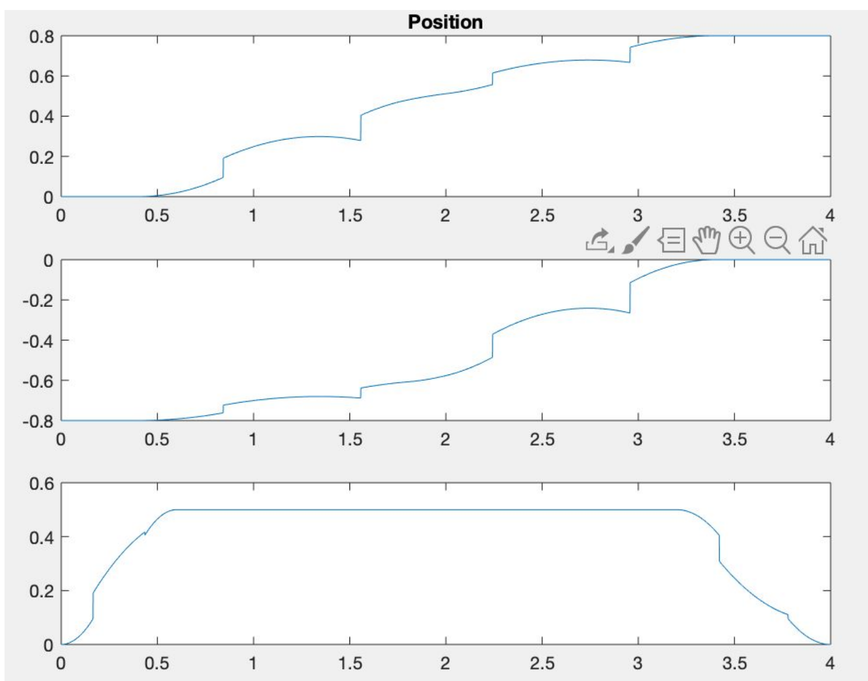


Figure. Plot of the position for every joint.

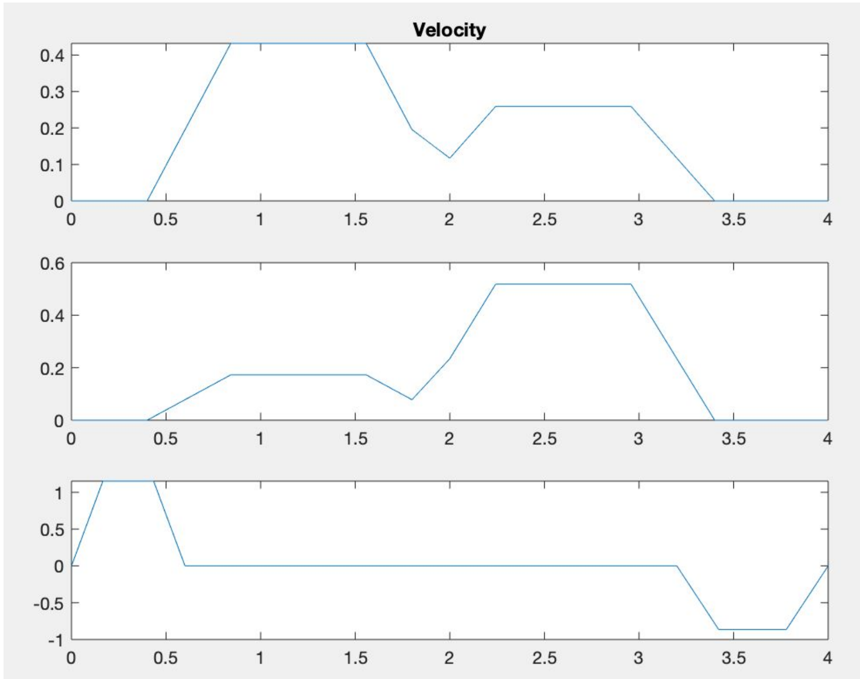


Figure. Plot of the velocity in every joint.

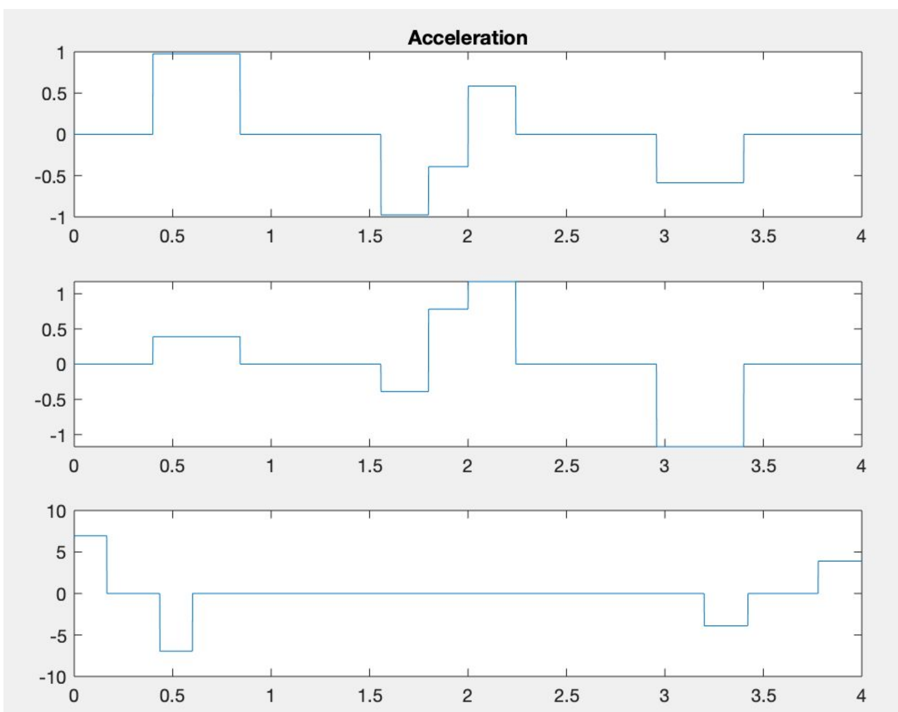


Figure. Plot of the acceleration in every joint.

Optional part

B_Hat function

The **b_hat** function is said to represent the adopted computational model in terms of estimates of the terms in the dynamic model. The error on the estimates, i.e., the uncertainty, is represented by

$$\widetilde{B} = \widehat{B} - B$$

This is due to imperfect model compensation as well as to intentional simplification in inverse dynamics computation. Notice that by setting B) = **B_hat**(where **B_hat** is the diagonal matrix of average inertia at the joint axes)
In our equation, the B_hat function is the same as the previous **b** function with just the (M_14 = 0).

Our equation is below:

```
B11 = ml1*l1^2 + Il1 + Im1 * kr1^2 + ml2*(0.25+l2^2+l1*l2) + Il2
      + Im2 + ml3 + Im3 + ml4*(0.5*(1+cos(th2))) + Il4;
B12 = ml2 * (l2^2 + 0.5*l2*cos(th2)) + Il2 + Im2*kr2 + ml3 + Im3
      + ml4*(0.25 * (1 + cos(th2))) + Il4;
B13 = -ml3 - Im3*kr3;
B14 = Il4;
B21 = ml2*(l2^2+0.5*l2*cos(th2)) + Il2 + Im2*kr2 + ml3 + Im3
      + ml4*(0.25*(1+cos(th2))) + Il4;
B22 = ml2*l2^2 + Il2 + Im2*kr2^2 + ml3 + Im3 + ml4*0.25 + Il4;
B23 = -ml3 - Im3*kr3;
B24 = Il4;
B31 = -ml3-Im3*kr3;
B32 = -ml3-Im3*kr3;
B33 = ml3 + Im3*kr3^2;
B34 = 0;
```

```
B41 = Il4;
B42 = Il4;
B43 = 0;
B44 = Im4;
OUT = [B11 B12 B13 B14
       B21 B22 B23 B24
       B31 B32 B33 B34
       B41 B42 B43 B44]
  + Iml4 * [1 1 0 kr4;
            1 1 0 kr4;
            0 0 0 0;
            kr4 kr4 0 kr4^2];
```

n_Hat function

The **n_hat** function is said to represent the adopted computational model in terms of estimates of the terms in the dynamic model. The error on the estimates, i.e., the uncertainty, is represented by

$$\tilde{n} = \hat{n} - n$$

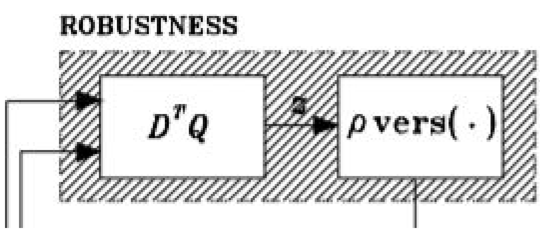
In our equation, the n_hat function is the same as the previous **b** function with just the (M_14 = 0)

Our calculation is below:

```
h = -1/2 * sin(th2) * (ml2 * l2 + ml4 * 0.5);

C = [h*th2_dot   2*h*th2_dot 0 0
     -h*th1_dot       0      0 0
     0 0 0 0
```

```
        0 0 0 0];

Fv = [0.0001 0 0 0
         0 0.0001 0 0
         0 0 0.01 0
         0 0 0 0.005];

g = [ml3*g ml3*g -ml3g 0];

n = C * joint_velocities + Fv * joint_velocities + g;
```

Robustness function



The control scheme is composed by 3 main parts。

- Compensation of nonlinear effects and decoupling。
- Feedforward and feedback action。
- Robust action

$$u = \widehat{B}(q)y + \widehat{n}(q, \dot{q})$$

$$\boldsymbol{w} = \begin{cases} \dfrac{\rho}{\|\boldsymbol{z}\|}\boldsymbol{z} & \text{per } \|\boldsymbol{z}\| \geq \epsilon \\ \dfrac{\rho}{\epsilon}\boldsymbol{z} & \text{per } \|\boldsymbol{z}\| < \epsilon \end{cases}$$

Explain using words the main disadvantages of this inverse dynamic control approach:

Pure inverse dynamic systems have the disadvantage that the main dynamic model that describes the system's manipulator is often not known with high precision.