UNIVERSITÀ DI BOLOGNA

School of Engineering

Master Degree in Automation Engineering

Industrial Robotics

Report assignment Matlab

Professors:
Claudio Melchiorri,
Gianluca Palli

Students:
Stefano Mulargia

Academic year 2020/2021

Contents

Chapter 1

Joint space trjectories

Compute and plot the joint space trajectory for a Puma 560 manipulator corresponding to a straight segment and a circle in the work space. Discuss and comment the situations in which the inverse kinematics service does not converge using the manipulability measures

1.1 Implementation

It is necessary to create the model of the Puma 560 by the command

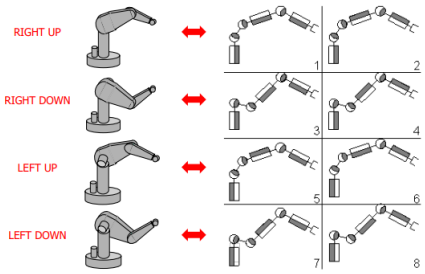
mdl_puma560 ;

It automatically generates the workspace variable p560 which describe the kinematic and dynamic characteristics of a Unimation Puma 560 model using the standard Denavit-Hartenberg conventions.

1.1.1 Linear Trajectory

In the Linear trajectory it is being used the **Function_trajLine** where are defined passed as parameter the starting point "*start_point*", destination point "*dest_point*", the model of the robot "*p560*", the number of step that it will use to compute the trajectory "*n_step*" and finally the singular configuration "*sing_cfg*" or "*sing_cfg_1*"

In general, the inverse kinematics for a Unimation Puma 560 leads to eight solution, two for each class named *right up*, *right down*, *left up*, *left down*,



In particular the Unimation Puma 560 has a wrist singularity when q_5 is equal to zero and the axes of joint 4 and 6 become aligned. The following section will shows the results.

The manipulability index measure the ability of a manipulator to move easily in any direction. In general, it can be possible that during the evolution of the motion, the manipulator passes through singular configuration.

1.1.2 Unimation Puma 560 with $q_5 = \pi/4$

Look at the behaviour of the robot.

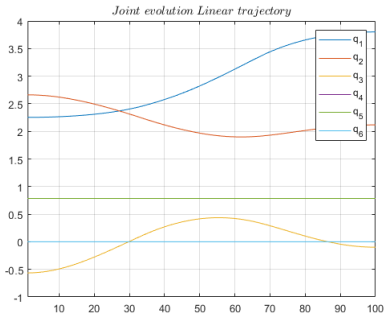


Figure 1.1: joint evolution singular configuration $q_5 = \pi/4$

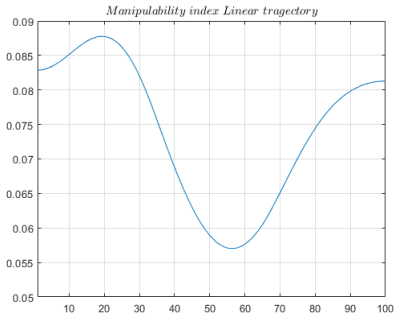


Figure 1.2: manipulability singular configuration $q_5 = \pi/4$

The manipulability is quite good but, as we can see in Figure 1.2. Around 55 we have a minimum which means that in this particular configuration at that moment, we had the smallest capability to the motion.

1.1.3 Wrist singularity Unimation Puma 560 $q_5 = 0$

In this experiment it is shown how if we have a singular configuration of the wrist the behaviour changes dramatically.

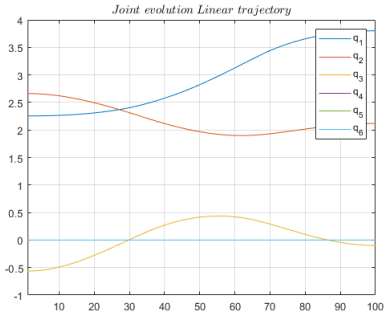


Figure 1.3: joint evolution singular configuration $q_5 = 0$

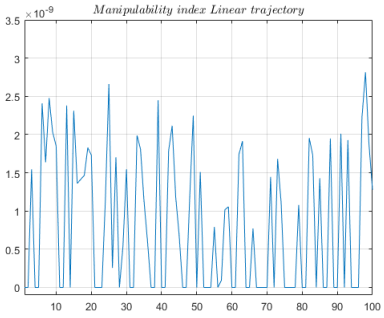


Figure 1.4: manipulability singular configuration $q_5 = 0$

Looking at Figure 1.4 we see how the manipulability index is almost zero in the whole trajectory. Note that all value are in the order of magnitude of 10^{-9}, so very small. This is due to the singular configuration of the wrist, where the q_4 and q_6 is aligned

1.1.4 Circular Trajectory

In the Circular trajectory problem it is being used the **Function_trajCircle** where are passed as parameter the radius "*radius*", the x-coordinate "*x_x*",the y-coordinate "*y_x*", the model of the robot "*p560*", the number of step that it will use to compute the trajectory "*n_step*" and finally the singular configuration "*sing_cfg*" or "*sing_cfg_1*".

In the following section it it shown the plot for the evolution of the joint in the workspace and the manipulability index for both configuration of the wrist. Note that $q_4 = q_6 = 0$ for both cases.

1.1.5 Unimation Puma 560 $q_5 = \pi/4$

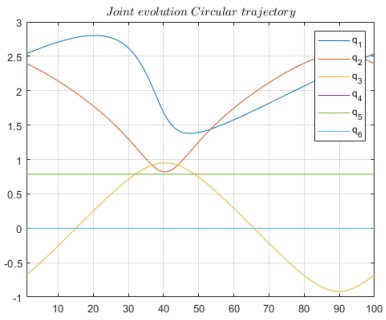


Figure 1.5: joint evolution singular configuration $q_5 = pi/4$

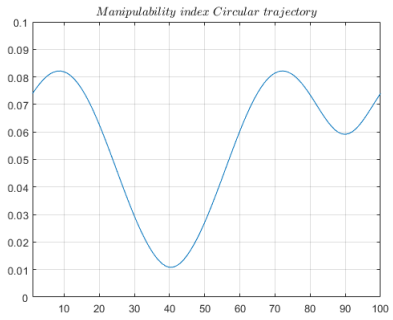


Figure 1.6: manipulability singular configuration $q_5 = pi/4$

In this case we notice, as for the linear trajectory, that the behaviour of the manipulability index is good.

1.1.6 Wrist singularity Unimation Puma 560 $q_5 = 0$

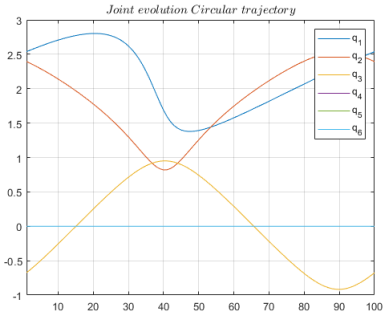


Figure 1.7: joint evolution singular configuration $q_5 = 0$

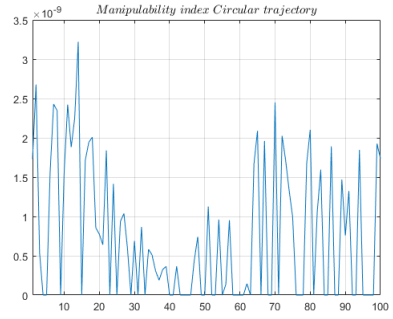


Figure 1.8: manipulability singular configuration $q_5 = 0$

As for the linear trajectory case that manipulability index in the case of the singular configuration of the wrist is very close to zero almost everywhere.

Note that the joint evolution is equal for both configuration.

Chapter 2

PD + gravity compensation

Implement a PD + gravity compensation controller for a Puma 560 manipulator and control the robot along the previously computed trajectory. Discuss the stability and tracking properties of the scheme in relation to different choices of the gains.

2.1 Problem formulation

This control technique is used in order to ensure that the manipulator reaches a desired configuration q_d.

Defining the model of the manipulator as:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = u$$

Defining the candidate Lyapunov function as

$$V(\tilde{q},\dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q} + \frac{1}{2}\tilde{q}^T K_p \tilde{q} > 0 \qquad \forall \dot{q}, \tilde{q} \neq 0$$

deriving we get

$$\dot{V} = -\dot{q}D\dot{q} + \dot{q}^T[u - g(q) - K_p\tilde{q}]$$

we define as control input the following function

$$u = g(q) + K_p\tilde{q} - K_d\dot{q}$$

Then we get

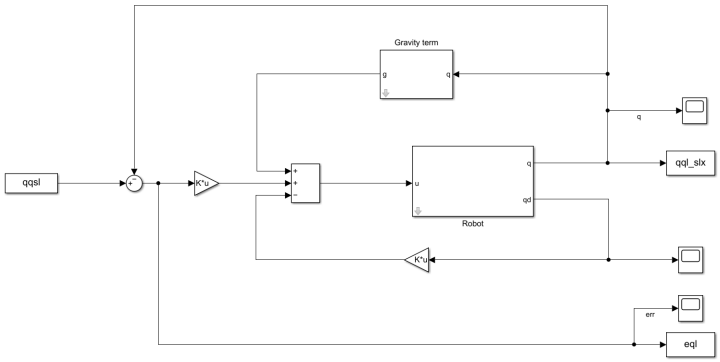
$$\dot{V} = \dot{q}^T(D + K_d)\dot{q}$$

where

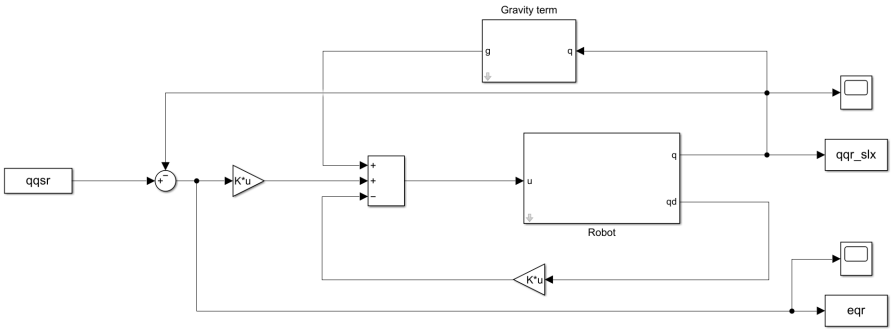
$$\tilde{q} = q_d - q$$

q_d is the desired position
q is the position of the robot
K_p is the proportional part of the controller
K_d is the derivative action of the controller

2.2 Simulink Scheme Linear trajectory



2.3 Simulink Scheme Circular trajectory



2.4 Stability and tracking properties

From the theory we know that, for constant q_d, given a perfect compensation of the gravity term, the system is globally asymptotically stable for any choice of K_p, K_d positive definite. Namely we should expect to reach the desired configuration without any error. In case we do not have the perfect compensation of the gravity term we end up with a steady state error.

Remember that we can see the PD controller as a spring-dumper system where the spring store is given by the K_p matrix and the "damping-effect" is controlled through the K_d matrix.

In the following example we will consider:
$K_p = diag\{\omega_{ni}^2\}$ and $K_d = diag\{2\omega_{ni}\delta_i\}$

2.5 Plot comparison Linear trajectory

2.5.1 case I

$$K_p = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad K_p = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

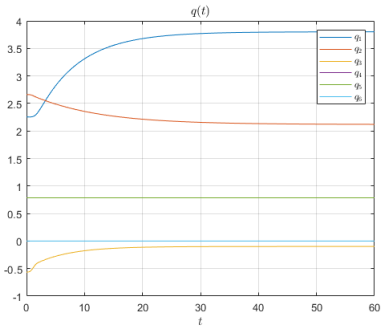


Figure 2.1: Joint evolution

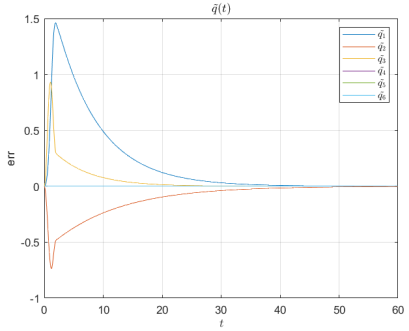


Figure 2.2: Error

In the above plots we see that even with matrices with low coefficient we reach zero error. It takes a lot of time, about fourty-five seconds, but it works.

2.5.2 case II

$$K_p = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \qquad K_p = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 \end{bmatrix}$$

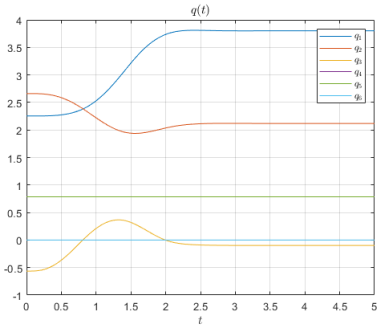


Figure 2.3: Joint evolution

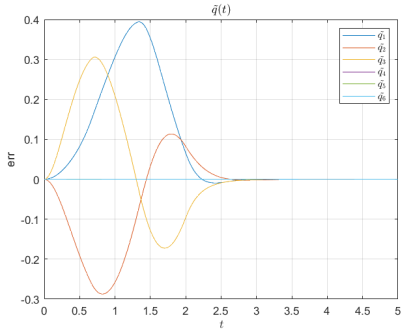


Figure 2.4: Error

Increasing the value of the matrix we get a faster stabilization, in particular we can appreciate that now the error goes to zero in three seconds whereas in "Case I" it tooks around fourty-five. Moreover we get a reduction also of the error of the joint.

2.5.3 case III

$$K_p = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \qquad K_p = \begin{bmatrix} 40 & 0 & 0 & 0 & 0 & 0 \\ 0 & 40 & 0 & 0 & 0 & 0 \\ 0 & 0 & 40 & 0 & 0 & 0 \\ 0 & 0 & 0 & 40 & 0 & 0 \\ 0 & 0 & 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 & 40 \end{bmatrix}$$

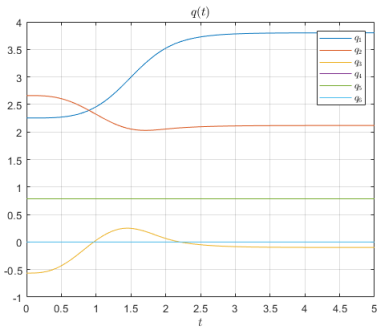


Figure 2.5: Joint evolution

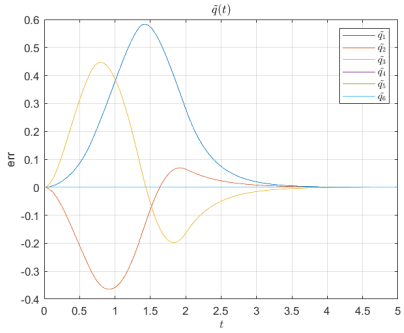
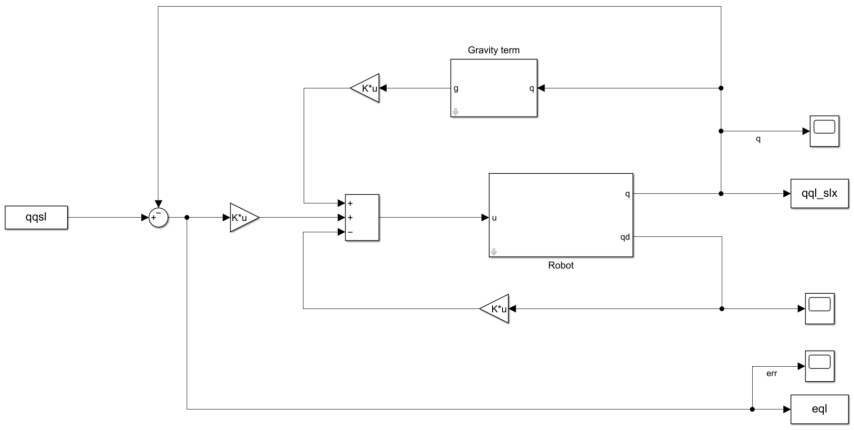


Figure 2.6: Error

In Case III, K_p is constant with respect to Case II, K_d changes. The error goes to zero a little bit slower so we can expect a slightly nervous dynamics of the error. Another consideration is that doubling the factor inside the matrix implies a increasing of the peak with the factor 1,5.

2.5.4 case no perfect compensation gravity term

In order to show the steady-state error due to a non-perfect compensation of the gravity term, in this example only 90% is compensated it is implemented previous control scheme with a slightly difference



$$K_p = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \qquad K_p = \begin{bmatrix} 40 & 0 & 0 & 0 & 0 & 0 \\ 0 & 40 & 0 & 0 & 0 & 0 \\ 0 & 0 & 40 & 0 & 0 & 0 \\ 0 & 0 & 0 & 40 & 0 & 0 \\ 0 & 0 & 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 & 40 \end{bmatrix}$$

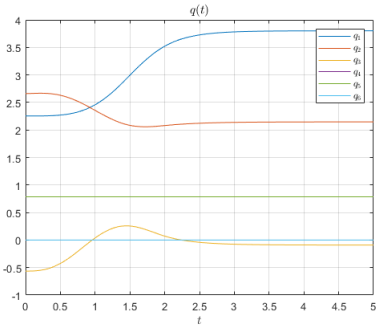


Figure 2.7: Joint evolution

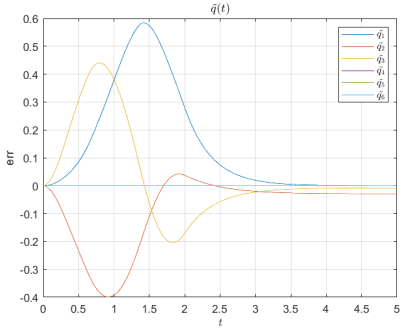


Figure 2.8: Error

As expected the errors does not vanishes, this is due to a not perfect compensation of the gravity term.

Chapter 3

Inverse Dynamics Control

Implement an Inverse Dynamics controller for a Puma 560 manipulator and steer the robot along a smooth joint space trajectory (e.g. computed with the "jtraj" block). Discuss the stability of the scheme if some parametric uncertainties are present (the model of a Puma 560 with parametric uncertainties will be provided)

3.1 Problem formulation

As suggested in order to compute the trajectory it is used the Matlab command

```
[qq, qqd, qqdd] = jtraj(q_init, q_final, T);
```

which, given as input the initial and final joint-space configuration *q_init* and *q_final*, computes position, velocity and acceleration profiles in T steps by using a fifth order polynomial trajectory with default zero boundary conditions for velocity and acceleration,in such way continuity in acceleration is achieved.

Since the model is as nonlinear MIMO defined as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + D\dot{q} + g(q) = u$$

we can see that the model is linear in the control input **u** and we know that the matrix **M(q)** is invertible for any configuration of the manipulator.

We can define a control input **u** based on the state feedback

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

this control input allows the feedback linearization computing the inverse dynamics in its inner loop.

$$\ddot{q} = y$$

The outer loop is used to stabilize the system with a PD controller

$$y = \ddot{q} + K_d\dot{\tilde{q}} + K_p\tilde{q}$$

3.2 Simulink scheme

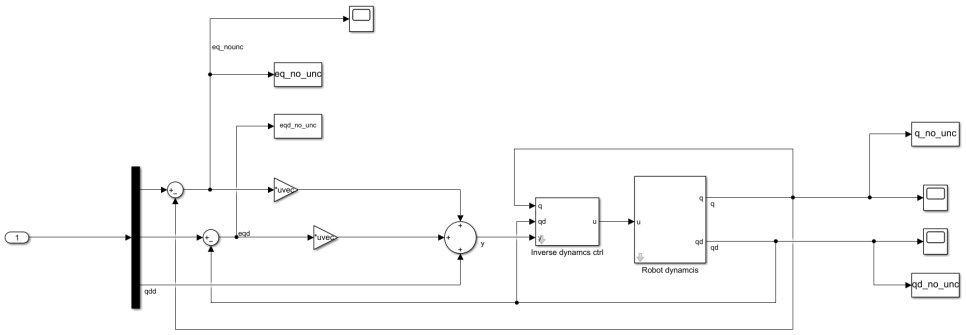


Figure 3.1: Simulink inverse dynamic scheme

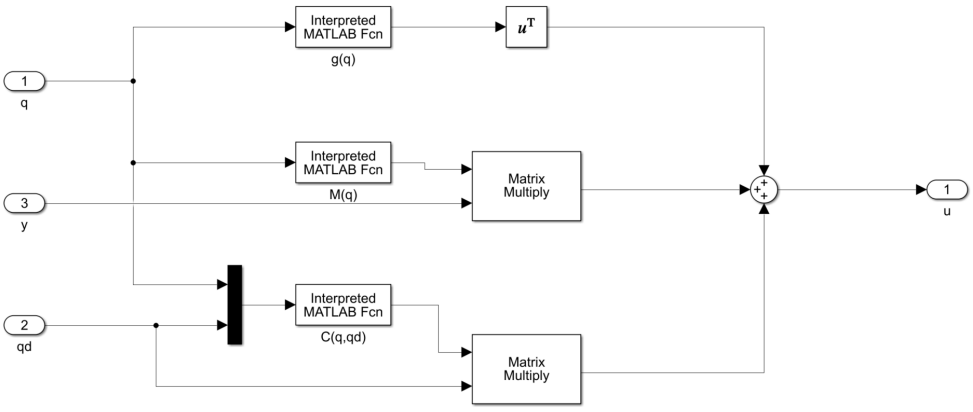


Figure 3.2: Simulink controller inverse dynamic scheme

3.3 Stability of the scheme with parametric uncertainties

Since K_p and K_d must be positive definite, it has been chosen:

$$K_p = \begin{bmatrix} 400 & 0 & 0 & 0 & 0 & 0 \\ 0 & 400 & 0 & 0 & 0 & 0 \\ 0 & 0 & 400 & 0 & 0 & 0 \\ 0 & 0 & 0 & 400 & 0 & 0 \\ 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 0 & 0 & 400 \end{bmatrix} \qquad K_d = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 20 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 0 & 0 & 0 \\ 0 & 0 & 0 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 20 \end{bmatrix}$$

It is performed a simulation along the trajectory, computed as previously described, which goes from q_{init} to q_{final}.

$$q_{init} = [0.0, 0.1, 0.0, 0.0, 0.1, 0.0]$$

$$q_{final} = [0.2, 0.3, 0.1, 0.4, 0.25, 0.35]$$

3.3.1 Ideal model

We can appreciate from the figure that every joint variable goes to the desired final set-point and the position error approaches the zero value after a transient.

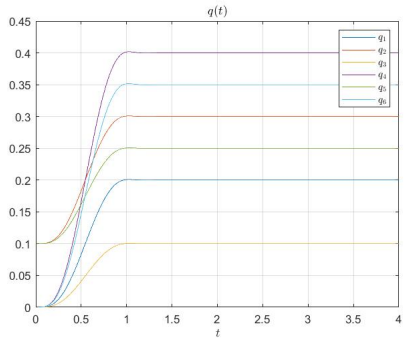


Figure 3.3: Joint evolution no uncertainties

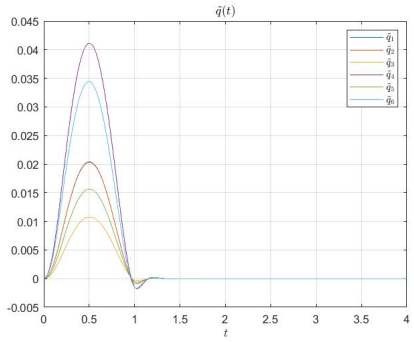


Figure 3.4: Position error no uncertainties

3.3.2 Uncertainties in the weight and in the inertia matrix of the second link

In the case with uncertainties we see a residual position error due to the non-perfect cancellation of the non-linear dynamics.

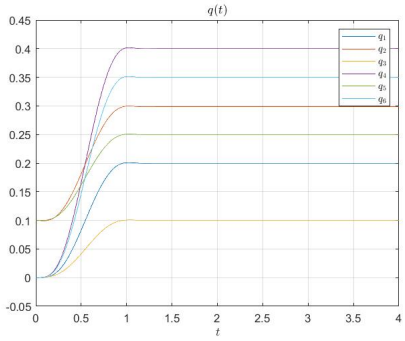


Figure 3.5: Joint evolution uncertainties

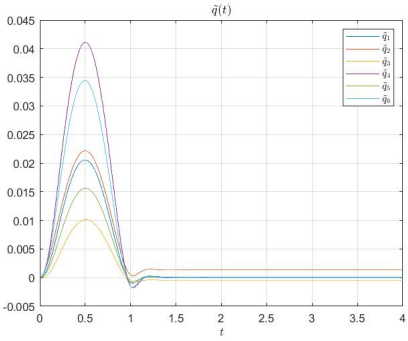


Figure 3.6: Position error uncertainties

Notice that there is also an error for links 1 and 3: this because, even if the uncertainty is only on link 2, due to the coupling effects among the joints also the others end up with a steady-state position error.

Chapter 4

Robust Sliding-Mode based Control

Implement a Robust Sliding-Mode based Controller able to stabilize the "uncertain" Puma 560 along the smooth joint space trajectory. Discuss the obtained performances with respect to variations of the parameters in the controller.

4.1 Problem formulation

In general, it is difficult to compensate exactly the dynamics of a robot. As it shown in the previous chapter the error does not go to zero with uncertainties in the model. A possible solution is to implement a robust control for industrial manipulator.

Let consider the inverse dynamics control

$$u = \hat{M}(q)y + \hat{n}(q, \dot{q})$$

where $\hat{M}$ and $\hat{n}$ represent the known part.

$$\ddot{q} = y + (M^{-1}\hat{M} - I)y + M^{-1}\tilde{n} = y + \eta$$

In order to compensate the uncertainty we can consider

$$y = \ddot{q_d} + K_d\dot{\tilde{q}_d} + K_p\tilde{q_d} + w$$

where the term w is added to compensate the uncertainties.

The term $\ddot{q_d} + K_d\dot{\tilde{q}_d} + K_p\tilde{q_d}$ aims to stabilizing the error dynamics, which is described as

$$\dot{\xi} = \tilde{H}\xi + D(\eta - w)$$

where

$$\tilde{H} = (H - DK) = \begin{bmatrix} 0 & I \\ -K_p & -K_d \end{bmatrix} \qquad D = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

Exploiting the Lyapunov method, we define as Lyapunov function

$$V(\xi) = \xi^T Q \xi$$

with Q a symmetric and positive definite matrix which is the unique solution of the Lyapunov equation

$$\tilde{H}^T Q + Q\tilde{H} = -P$$

Then, the derivative of the Lyapunov function is

$$\dot{V} = -\xi^T P \xi + 2z^T(\eta - w)$$

where

$$z = D^T Q \xi$$

By choosing

$$w = \frac{\rho}{||z||} z$$

we have that $\dot{V} < 0$ for any choice of the gain $\rho \geq ||\eta|| \qquad \forall q, \dot{q}, \ddot{q}$.

This term guarantee robustness, since it compensate for uncertainties. A vectorial control action is obtained. This control action guarantees that in steady state $\xi = \dot{\xi} = 0$ then $\tilde{q} = \dot{\tilde{q}} = 0$- Moreover, the trajectories in the ξ space converge to the *Sliding subspace* defined by

$$S(\xi) = z = D^T Q \xi = 0$$

The implementation of this method, also known as *Variable Structure Control*, has the side effect of the chattering phenomenon in which oscillations are generated in the controlled system because once we reach the sliding region, we start commuting the control action between its two maximum and minimum values. This is quite bad because this kind of action can excite non-modelled dynamics and undesired behaviours may be obtained, e.g. oscillations.

chattering phenomenon in which oscillation are generated in the controlled system due to non modelled dynamics that may be excited. In general it is already present but can be mitigate by the **boundary layers** methods

$$w = \begin{cases} \rho \frac{z}{||z||}, & \text{per } ||z|| \geq \varepsilon \\ \\ \rho \frac{z}{\varepsilon}, & \text{per } ||z|| \leq \varepsilon \end{cases}$$

4.2 Simulink scheme

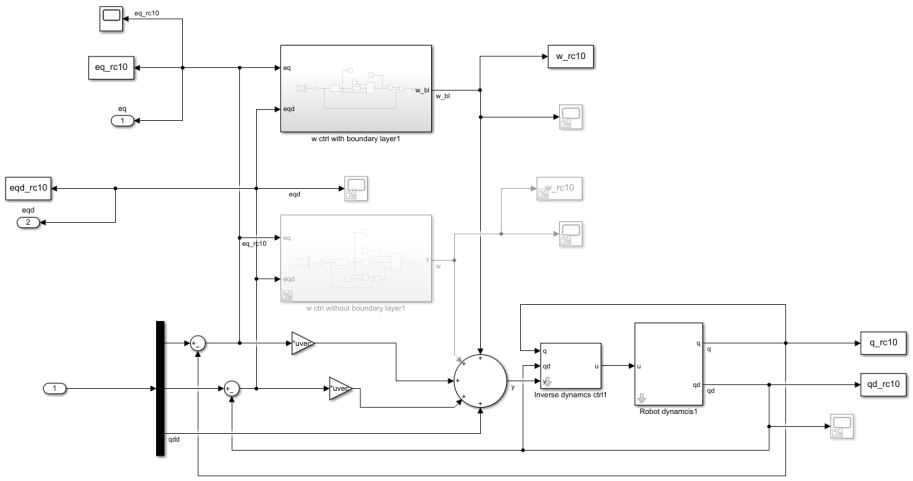


Figure 4.1: Simulink robust control scheme

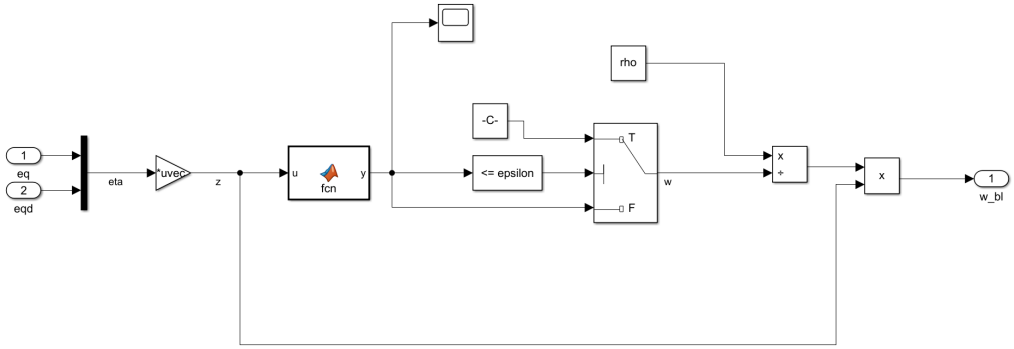


Figure 4.2: Simulink compensation control scheme

4.3 Implementation

In the following section will be shown different cases, here are reported the data common for all experiment.

$$q\_\{init\} = [0.0,\ 0.1,\ 0.0,\ 0.0,\ 0.1,\ 0.0\ ]$$

$$q\_\{final\} = [0.2,\ 0.3,\ 0.1,\ 0.4,\ 0.25,\ 0.35\ ]$$

In the simulations I defined $\rho = 20$ and the matrix P as diagonal

$$P\_1 = \mathbf{diag}\{10\ \ 100\ \ 10\ \ 1\ \ 1\ \ 1\ \ 0.001\ \ 0.001\ \ 0.001\ \ 0.01\ \ 0.01\ \ 0.01\}$$

4.4 Stability of the scheme with parametric uncertainties

4.5 Z_1 without boundary layers

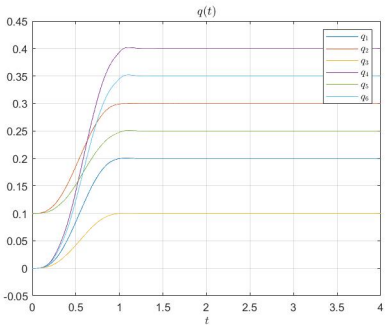


Figure 4.3: Joint evolution

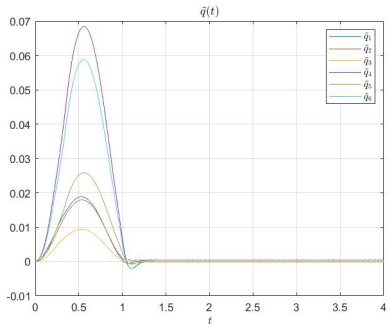


Figure 4.4: Joint position error

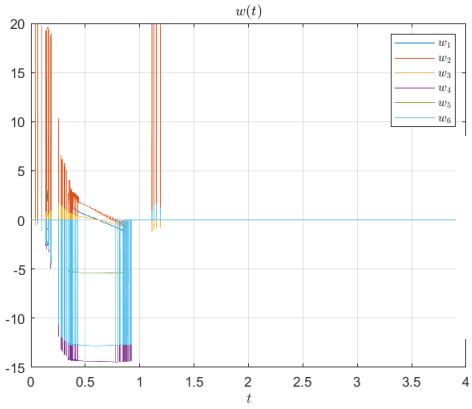


Figure 4.5: Wrench

We can look at how the chattering phenomena appear in the control input, make it pass through the maximum and the minimum value very quickly.

4.6 Z_1 with boundary layers

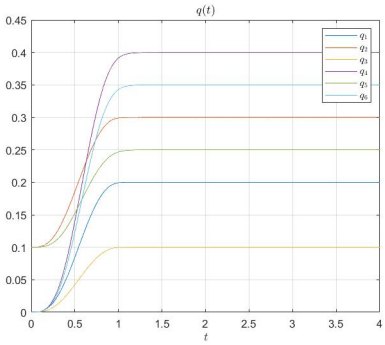


Figure 4.6: Joint evolution

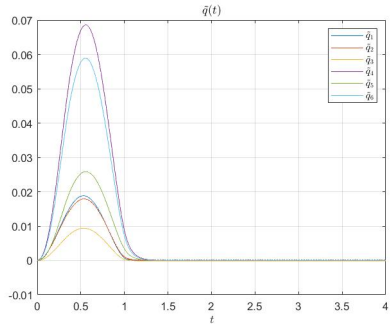


Figure 4.7: Joint position error

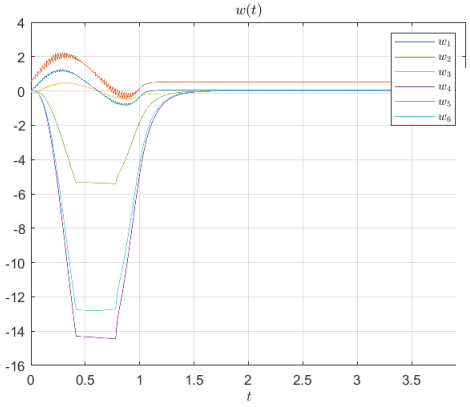


Figure 4.8: Wrench

Notice that with boundary layer we have a smoother behaviour with respect the case in which I do not have that boundaries.

4.7 Z_{10} without boundary layers

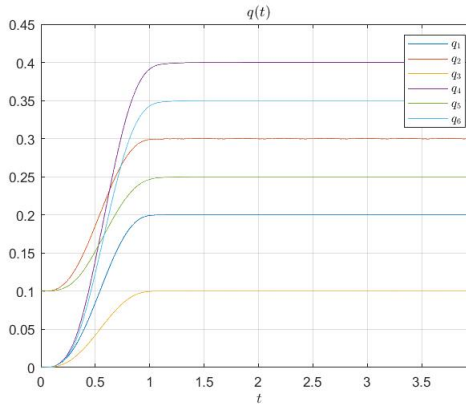


Figure 4.9: Joint evolution position

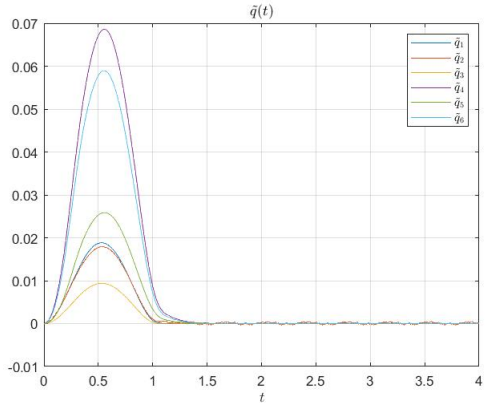


Figure 4.10: joint position Error

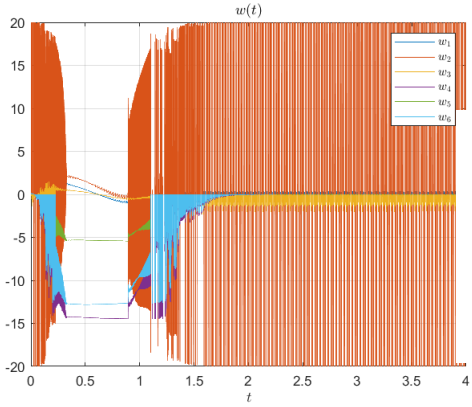


Figure 4.11: Wrench

Looking at figure 4.11 we can see how the control input $w(t)$ has a behaviour where, when the error is small the input oscillates between -20 and +20 very quickly.

4.8 Z_{10} with boundary layers

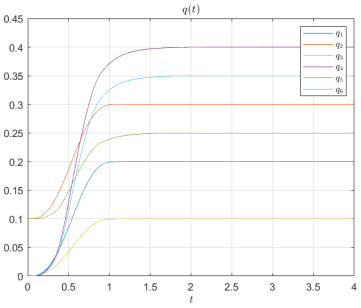


Figure 4.12: Joint evolution

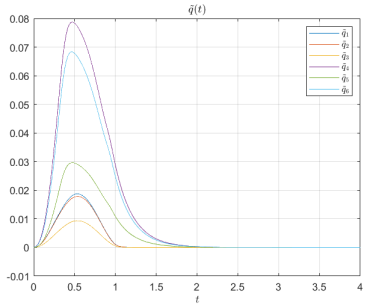


Figure 4.13: Joint Position error

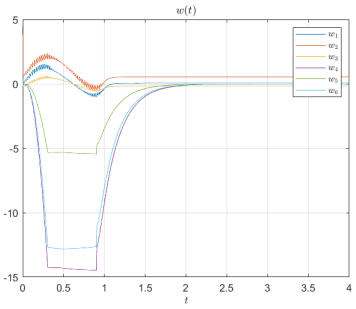


Figure 4.14: Wrench

In this case we can appreciate how we can avoid the previous behaviour with the introduction of the boundary layers.

Chapter 5

PD + gravity Compensation in the workspace

Es 5

Implement a PD + gravity compensation or an Inverse Dynamics controller in the workspace. Discuss the difference from a conceptual and computational point of view with respect to the corresponding joint space controller.

Es 6

With reference to the previous problem, implements also the tool orientation control in the workspace

5.1 Problem formulation

Often is convenient to express the trajectories in the workspace, in order to implement this feature we have to work with the inverse kinematic of our model to translate the motion specification in the joint space. Usually the computation frequency of the trajectory is lower then the one of the control system, we have to interpolate it. Note that this interpolation implies a delay in the input of the joint coordinate in the controlled system, this delay is function also of the order is the interpolating function for this reason we will use linear interpolating function.

Since the system will plan the trajectory in work space but i always control the system in the joint space. from the point of view of the x-variables we are in open-loop. It is necessary to have a precise kinematic model otherwise I will make an error during the Inverse Kinematic transformation.

5.2 Scheme

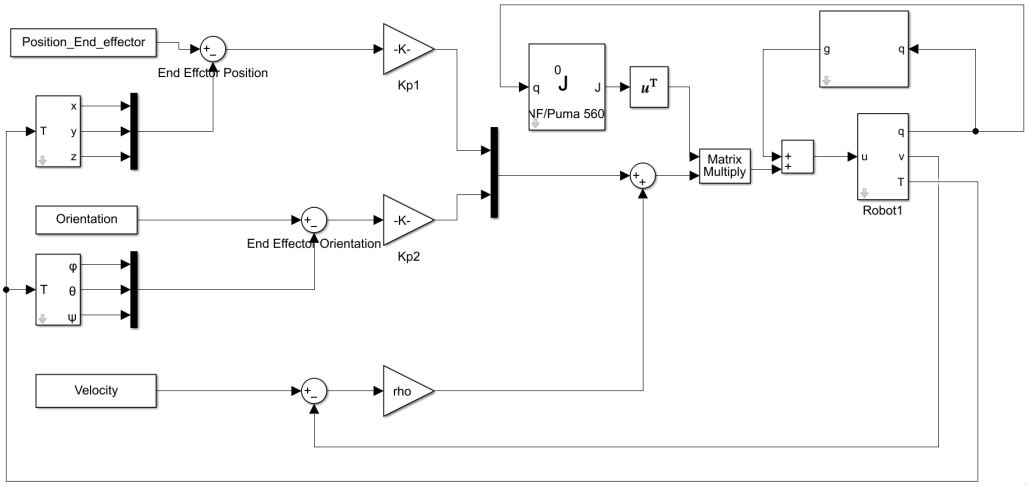


Figure 5.1: PD + gravity compensation in Workspace

5.3 Comments about comparison between joint space and workspace control scheme

Compared to a joint space control scheme, a workspace control scheme is often less performant since it implement in the control loop heavy computation that are somewhat representative of the inversion of the dynamic of the robot that can require higher sampling time. Nonetheless, workspace control schemes are necessary when dealing with interactions with the environment, since if we want to control both position and interaction forces (such as in an elastic environment) we need to implement these workspace schemes.

5.4 Plots

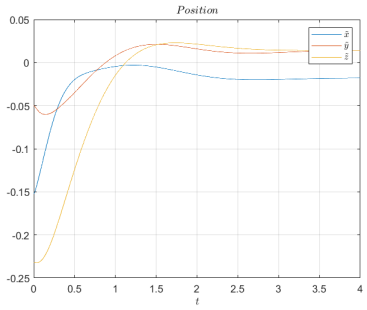


Figure 5.2: Position error end-effector

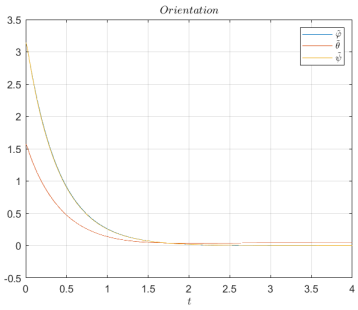


Figure 5.3: Orientation error end-effector

Chapter 6

Trajectory on a known surface with fixed distance and limited extension

Suppose that, in order to kill the coronavirus, the robot is equipped at the wrist interface with an UV lamp able to spread light over a pyramid with aperture of 30 degrees point out of the wrist center along the 6-th joint z-axis. The student is asked to design a robot trajectory able to scan a planar or (a portion of) cylindrical surface of known shape and limited extension (reachable by the robot) maintaining a fixed distance and orientation of the UV lamp and the surface to scan, ensuring that the whole surface is covered by the UV light.

6.1 Planar Surface

Since our goal is to irradiate the whole surface with an UV lamp keeping the robot a constant distance. Knowing position and orientation of the plane in space, it is possible to compute the normalized plane equation $ax + by + cz = 1$. This give us the direction of the vector perpendicular to the plane $\hat{n} = \frac{[abc]^T}{||[abc]^T||}$. This will be used as the z axis.

First we have to define the plane. We define the point of the square plane and then we project it in the space shifting it to keep the center of the lamp inside the plane.

Now we define the "extremities" which are the corners of the plane at the chosen distance from the surface. It has been obtained aligning the base of the pyramid with the planar surface and scanning over it so that the base never exit the plane.

Finally we can use the *ctraj* function to compute the trajectory.