Mattia Piras 4524072 06/2022

Robotics Engineering AY. 2021/2022

Università di Genova

Robot dynamics and control third assignment's report

Control problem for a manipulator

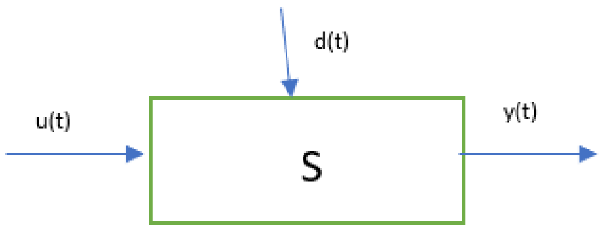
Introduction:

The aim of this assignment is to control a UR5 robot manipulator in such a way to have some desired characteristics.

But first thing first let me introduce in a better way the problem of control:

Starting from the well-known equation [1] we want to control my system S considering disturbances:

[1] $A(q)\ddot{q} + B(q,\dot{q})\dot{q} + C(q) = \tau$



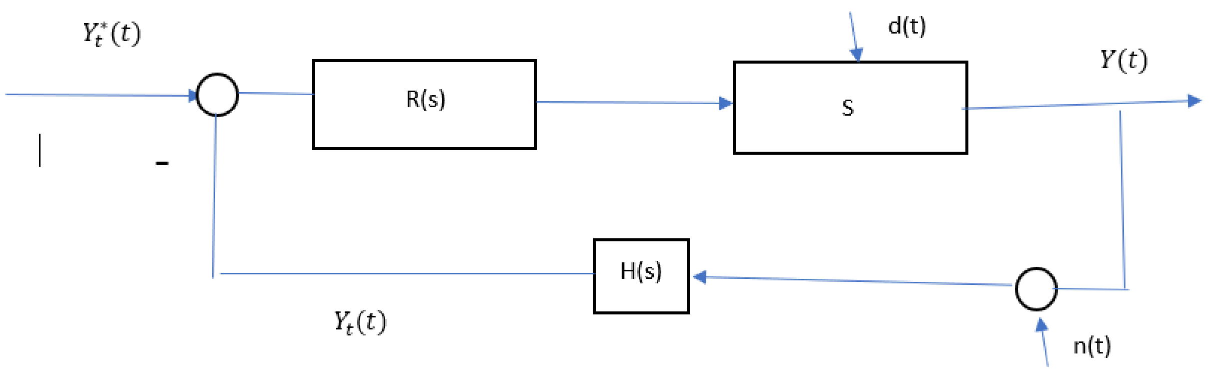
What's important in this case is to check:

1) Stability
2) Precision
3) Disturbances Rejection (How to behave when disturbances occur)
4) Robustness (the control system must be capable ensuring 1,2,3 despite the changing in the system)

The possibility of Computing the control signal u(t) s.t these four specifications are satisfied is important to have $y(t) \approx y^*(t)$ most of the time.

- In presence of disturbances
- In presence of uncertainties in model of the system

In order to have a better control system feedback is needed. Knowing my output during time allow me to change my control on the base on where I want to arrive or what I want to achieve.



NOTE: We need now to move from ANALOG Domain To the Digital one, but I'll not enter in this particular case.

Robot Control Basics

The main challenge in the motion control problem of rigid manipulators is the complexity of their dynamics and uncertainties. The former results from nonlinearity and coupling in the robot manipulators. The latter is twofold: structured and unstructured. Structured uncertainty means imprecise knowledge of the dynamic parameters, whereas unstructured uncertainty results from joint and link flexibility, actuator dynamics, friction, sensor noise, and unknown environment dynamics.

Introduction to Motion Control

The dynamical model of robot manipulators will be recalled in this section. Furthermore, important properties of this dynamical model, which is useful in controller design, will then be addressed. Finally, different control tasks of the robot manipulators will be defined.

Dynamical Model For motion control

the dynamical model of rigid robot manipulators is conveniently described by Lagrange dynamics. Let the robot manipulator have n links and let the (n × 1)- vector q of joint variables be $q = [q1, \dots, qn]^T$.

The dynamic model of the robot manipulator is then described by Lagrange’s equation:

$$A(q)\ddot{q} + B(q, \dot{q})\,\dot{q} + C(q) = \tau$$

Where:

- $A(q)$ is the (n × n) inertia matrix
- $B(q, \dot{q})\,\dot{q}$ is the (n × 1)-vector of Coriolis and centrifugal forces
- $C(q)$ is the (n × 1)-vector of gravity force
- τ is the (n × 1) -vector of joint control inputs to be designed.

Friction and disturbance input have been neglected here.

Remark:

Other contributions to the dynamic description of the robot manipulators may include the dynamics of the actuators, joint and link flexibility, friction, noise, and disturbances. Without loss of generality, the case of the rigid robot manipulators is stressed here. The control schemes that I will introduce after are based on some important properties of the dynamical model of robot manipulators. Before giving a detailed introduction to these different schemes, let me first give a list of those properties.

Property 1.1

The inertia matrix is a symmetric positive-definite matrix, which can be expressed

$\lambda_h I_n \leq$ A(q) $\leq \lambda_H I_n$ where λ_h and λ_H denote positive constants.

Property 1.2

The matrix N(q, $\dot{q}$) = $\dot{A}$ (q) – 2B(q, $\dot{q}$) is skew symmetric for a particular choice of B(q, , $\dot{q}$) (which is always possible), i. e.,

z^TN(q, , $\dot{q}$)z = 0 for any (n × 1)-vector z.

Property 1.3

The (n ×n)-matrix B(q, $\dot{q}$) satisfies || B(q, $\dot{q}$) || ≤ b_o || $\dot{q}$ || for some bounded constant b_o

Property 1.4

The gravity force/torque vector satisfies ||C(q) ||≤ c_o for some bounded constants c_o.

Property 1.5

The equation of motion is linear in the inertia parameters. In other words, there is a (r × 1) constant vector a and an (n ×r) regressor matrix Y(q, $\dot{q}$, $\ddot{q}$) such that

$$A(q)\,\ddot{q} + B(q, \dot{q})\,\dot{q} + C(q) = Y(q, \dot{q}, \ddot{q})a$$

The vector a is comprised of link masses, moments of inertia, and the link, in various combinations.

Property 1.6

The mapping τ → $\dot{q}$ is passive; i. e., there exists α ≥ 0 such that

$\int_0^t \dot{q}^T(\beta)\tau(\beta)\, d\beta \geq -\alpha, \forall t < \infty$

Control Tasks

It is instructive for comparative purposes to classify control objectives into the following two classes:

- **Trajectory tracking** is aimed at following a timevarying joint reference trajectory specified within the manipulator workspace. In general, this desired trajectory is assumed to comply with the actuators' capacity. In other words, the joint velocity and acceleration associated with the desired trajectory should not violate, respectively, the velocity and acceleration limit of the manipulator. In practice, the capacity of actuators is set by torque limits, which result in bounds on the acceleration that are complex and state dependent.

- **Regulation** sometimes is also called point-to-point control. A fixed configuration in the joint space is specified; the objective is to bring to and keep the joint variable at the desired position in spite of torque disturbances and independently of the initial conditions. The behaviour of transients and overshooting, are in general, not guaranteed. The selection of the controller may depend on the type of task to be performed. For example, tasks only requiring the manipulator to move from one position to another without requiring significant precision during the motion between these two points can be solved by regulators, whereas such as welding, painting, and so on, require tracking controllers.

Joint Space Versus Operational Space Control

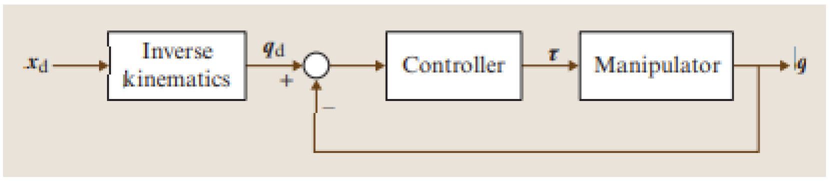
In a motion control problem, the manipulator moves to a position to pick up an object, transports that object to another location, and deposits it. Such a task is an integral part of any higher-level manipulation tasks such as painting or spot-welding. Tasks are usually specified in the task space in terms of a desired trajectory of the end-effector, while control actions are performed in the joint space to achieve the desired goals. This fact naturally leads to two kinds of general control methods, namely joint space control and operational space control (task space control) schemes

Joint Space Control

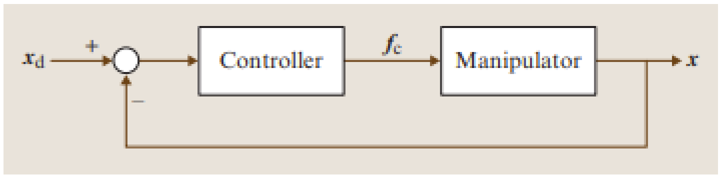
The main goal of the joint space control is to design a feedback controller such that the joint coordinates q(t) ∈ R n track the desired motion qd(t) as closely as possible. To this end, consider the equations of motion of an n-DOF manipulator expressed in the joint space. In this case, the control of robot manipulators is naturally achieved in the joint space, since the control inputs are the joint torques. Nevertheless, the user specifies a motion in terms of end-effector coordinates, and thus it is necessary to understand the following strategy. Figure below shows the basic outline of the joint space control methods. Firstly, the desired motion, which is described in terms of end-effector coordinates, is converted to a corresponding joint trajectory using the inverse kinematics of the manipulator. Then the feedback controller determines the joint torque necessary to move the manipulator along the desired trajectory specified in joint coordinates starting from measurements of the current joint states .

Since often the desired task is given in terms of the time sequence of the joint motion, joint space control schemes are quite adequate in situations where manipulator tasks can be accurately preplanned and little or no online trajectory adjustments are necessary

Typically, inverse kinematics is performed for some intermediate task points, and the joint trajectory is interpolated using the intermediate joint solutions. Although the command trajectory consists of straight-line motions in end-effector coordinates between interpolation points, the resulting joint motion consists of curvilinear segments that match the desired end-effector trajectory at the interpolation points.



Generic concept of joint space control



Basic concept of operational space

In fact, the joint space control includes simple PD control, PID control, inverse dynamic control as explained in the following sections.

Operational Space Control

In more complicated and less certain environments, end effector motion may be subject to online modifications in order to accommodate unexpected events or to respond to sensor inputs. There are a variety of tasks in manufacturing where these type of control problems arise. It is essential when controlling the interaction between the manipulator and environment is of concern. Since the desired task is often specified in the operational space and requires precise control of the end-effector motion, joint space control schemes are not suitable in these situations. This motivated a different approach, which can develop control schemes directly based on the dynamics expressed in the operational space.

Let us suppose that the Jacobian matrix, denoted by J(q) ∈ R n×n, transforms the joint velocity ($\dot{q} \in R^n$) to the task velocity ($\dot{x} \in R^n$) according to

$$\dot{x} = J(q)\,\dot{q}$$

Furthermore, assume that it is invertible. Then, the operational space dynamics is expressed as follows:

$$f_c = A(q)\,\dot{x} + B(q, \dot{q})\,\dot{x} + C(q)$$

where fc ∈ R^n denotes the command forces in the operational space.

the pseudo-inertia matrix is defined by

$$A(q) = J^{T(}q)\,H(q)\,\dot{J}(q) \qquad [0.1]$$

and B (q, $\dot{q}$) and C(q) are given by

$$B(q, \dot{q})\,\dot{q} = J^T(q)\,H(q)\,\dot{J}(q) + J^T(q)N(q, \dot{q}) \qquad [0.2]$$

$$C(q) = J^T(q)\,\tau_g(q) \qquad [0.3]$$

Recall

$$H(q) = \begin{bmatrix} I_{Ci} & 0 \\ 0 & m_i 1 \end{bmatrix}$$

$$N(q, \dot{q}) = [\begin{matrix} \omega_{\frac{i}{o}} \wedge I_{Ci}\,(\omega_{\frac{i}{o}}) \\ 0 \end{matrix}]$$

The task space variables are usually reconstructed from the joint space variables, via the kinematic mappings. In fact, it is quite rare to have sensors to directly measure end-effector positions and velocities. Also, it is worth remarking that an analytical Jacobian is utilized since the control schemes operate directly on task space quantities, i. e., the end-effector position and orientation. The main goal of the operational space control is to design a feedback controller that allows execution of an end-effector motion x(t) ∈ R^n that tracks the desired end-effector motion x_d (t) as closely as possible.

To this end, consider the equations of motion of the manipulator expressed in the operational space. For this case shows a schematic diagram of the operational space control methods. There are several advantages to such an approach because operational space controllers employ a feedback loop that directly minimizes task errors. Inverse kinematics need not be calculated explicitly since the control algorithm embeds the velocity-level forward kinematics and that's why I chose it in my third exercise.

PID Control

Traditionally, control design in robot manipulators can be understood as the simple fact of tuning of a PD or PID compensator at the level of each motor driving the manipulator joints. Fundamentally, a PD controller is a position and velocity feedback that has good closed-loop properties when applied to a double integrator system.

Actually, the strong point of PID control lies in its simplicity and clear physical meaning. Simple control is preferable to complex control, at least in industry, if the performance enhancement obtained by using complex control is not significant enough.

The physical meanings of PID control are as follows:

- P-control means the present effort making a present state into desired state;
- I-control means the accumulated effort using the experience information of previous states;
- D-control means the predictive effort reflecting the information about trends in future states.

PD Control for Regulation (WE USED THIS)

A simple design method for manipulator control is to utilize a linear control scheme based on the linearization of the system about an operating point. An example of this method is a PD control with a gravity compensation scheme.

Gravity compensation acts as a bias correction, compensating only for the amount of forces that create overshooting and an asymmetric transient behaviour. Formally, it has the following form:

$$\tau = K_p(q_d - q \quad) - K_v\,\dot{q} + \tau_g(q)$$

where K_p and $K_v \in R^{nxn}$are positive-definite gain matrices.

This controller is very useful for set-point regulation, i. e., q_d =constant

When this controller is applied to [1], the closed-loop equation becomes

$$A(q)\,\ddot{q} + B(q, \dot{q})\,\dot{q} + K_v\,\dot{q} - K_p\,e_q = 0 \quad [2]$$

Where:

$e_q = q_d - q$

and the equilibrium point is

$y = [\, e_q{}^{T} \,,\ \dot{q}^{T}]^{T} = 0$

Now, the stability achieved by PD control with gravity compensation can be analysed according to the closed-loop dynamic [2].

Consider the positive-definite function V = $\frac{1}{2}\ \dot{q}^T$ A(q) $\dot{q}\ + \frac{1}{2}\ e_q{}^T K_v\ e_q$.

Then, the derivative of function becomes negative semidefinite for any value of $\dot{q}$ by using Property 1.2 previously mentioned

$$\dot{V}\ = - \dot{q}^T\ K_v\ \dot{q}\ \leq\ - \lambda_{min}\ (K_v) || \dot{q} ||^2 \qquad [3]$$

where $\lambda_{min}\ (K_v)$ means the smallest eigenvalue of K_v. By invoking the Lyapunov stability theory and LaSalle's theorem [1.1], it can be shown that the regulation error will converge asymptotically to zero, while their high-order derivatives remain bounded. This controller requires knowledge of the gravity components (structure and parameters), though it is simple.

Now, consider simple PD control without gravity compensation

τ =$K_p\ (q_d\ - q) - K_v\ \dot{q}$

then the closed-loop dynamic equation becomes

$$A(q)\ \ddot{q} + B(q,\ \dot{q}\ )\dot{q}\ \ + \tau_v\ (q) + K_v\ \dot{q}\ \ - K_p\ e_q = 0\ . \qquad [4]$$

Consider the positive definite function

$$V = \frac{1}{2}\ \dot{q}^T\ A(q)\ \dot{q} + \frac{1}{2}\ e_q{}^T K_v\ e_q + U(q) + U_0\ \ ,$$

where U(q) denotes the potential energy with the relation of ∂U(q)/∂q = C(q) and U0 is a suitable constant.

Taking time derivative of V along the closed-loop dynamics [4] gives the same result [3] with previous one using gravity compensation. In this case, the control system must be stable in the sense of Lyapunov, but it can not conclude that the regulation error will converge to zero by LaSalle's theorem [1.1].

Actually, the system precision (the size of the regulation error vector) will depend on the size of the gain matrix K_p in the following form:

$||e_q|| \leq ||K_p{}^{-1}||\ g_0$

where g0 is that in Property 1.4. Hence, the regulation error can be arbitrarily reduced by increasing K_p; nevertheless, measurement noise and other unmodeled dynamics, such as actuator friction, will limit the use of high gains in practice.

Tracking Control

While independent PD controls are adequate in most set-point regulation problems, there are many tasks that require effective trajectory tracking capabilities.

In this case, employing local schemes requires moving slowly through a number of intermediate set points, thus considerably delaying the completion of the task. Therefore, to improve the trajectory tracking performance, controllers should take account of the manipulator dynamic model via a computed-torque-like technique. The tracking control problem in the joint or task space consists of following a given time-varying trajectory q_d(t) or x_d(t) and its successive derivatives $\ddot{q}_d(t)$ or $\ddot{x}_d(t)$ and $\dot{q}_d(t)$ or and $\dot{x}_d(t)$ which describe the desired acceleration and velocity, respectively.

Among the control approaches reported in the literature, typical methods include inverse dynamic control, the feedback linearization technique, and others

Inverse Dynamics Control

Though the inverse dynamics control has a theoretical background, such as the theory of feedback linearization discussed later, its starting point is mechanical engineering intuition based on cancelling nonlinear terms and decoupling the dynamics of each link. Inverse dynamics control in joint space has the form

$$\tau = A(q)v(t)+B(q, \dot{q})\,\dot{q} + C(q)$$

which, applied to [1], yields a set of n decoupled linear systems, e.g., $\ddot{q}$ = v, where v is an auxiliary control input to be designed. A typical choice for v is

$$v = \ddot{q}_d + K_v(\dot{q}_d - \dot{q}) + K_p\,(q_d - q)$$

leading to the error dynamics equation $\ddot{e}_q + K_v\dot{e}_q + K_p\,e_q = 0$

Alternatively, inverse dynamics control can be described in the operational space. Consider the operational space dynamics (6.9). If the following inverse dynamics control is used in the operational space, fc = A(q)$(\ddot{x}_d + K_v\dot{e}_x + K_p\,e_x)$+B (q, $\dot{q}$) $\dot{x}$ +C(q) ,

where $e_x = x_d - x$

the resulting error dynamics is

$$\ddot{e}_x + K_v\dot{e}_x + K_p\,e_x = 0$$

and it is also exponentially stable. One apparent advantage of using this controller is that K_p and K_v can be selected with a clear physical meaning in operational space. However, as can be seen in [0.1], A(q) becomes very large when the robot approaches singular configurations (look at the rule $\dot{x}$ = J(q) $\dot{q}$).

This means that large forces in some directions are needed to move the arm.

Feedback Linearization

This approach generalizes the concept of inverse dynamics of rigid manipulators. The basic idea of feedback linearization is to construct a transformation as a so-called inner-loop control, which exactly linearizes the nonlinear system after a suitable state space change of coordinates.

One can then design a second stage or outer-loop control in the new coordinates to satisfy the traditional control design specifications such as tracking, disturbance rejection etc…

The full power of the feedback linearization scheme for manipulator control becomes apparent if one includes in the dynamic description of the manipulator the transmission dynamics, such as the elasticity resulting from shaft windup, gear elasticity, etc.

Let us now develop a simple approach to the determination of linear state-space representations of the manipulator dynamics [1] by considering a general sort of output ξ ∈ R^p:

$$\xi = h(q)+r(t)$$

where

- h(q) is a general predetermined function of the joint coordinate q ∈ R^n
- r(t) is a general predetermined time function.

The control objective will be to select the joint torque inputs τ in order to make the output ξ(t) go to zero. The choice of h(q) and r(t) is based on the control purpose.

For example, if **h(q) = −q and r(t) = q_d (t),**

the desired joint space trajectory we would like the manipulator to follow, then

ξ(t) = q_d (t)−q(t) ≡e_q (t) is the joint space tracking error. Forcing ξ(t) to zero in this case would cause the joint variables q(t) to track their desired values q_d (t), resulting in a manipulator trajectory-following problem.

As another example, ξ(t) could represent the operational space tracking error

$$\xi(t) = x_d(t) - x(t) \equiv e_x(t).$$

Then, controlling ξ(t) to zero would result in trajectory following directly in operational space where the desired motion is usually specified.

To determine a linear state-variable model for manipulator controller design, let us simply differentiate the output ξ(t) twice to obtain

$\dot{\xi}$ = ∂h ∂q $\dot{q}$ +$\dot{r}$ = T$\dot{q}$ +$\dot{r}$

$\ddot{\xi}$ = T$\ddot{q}$ + $\dot{T}$ $\dot{q}$ + $\ddot{r}$

where we defined a (p×n) transformation matrix of the form

T(q) = ∂h(q) ∂q = µ ∂h ∂q_1 ∂h ∂q_2 · · · ∂h ∂q_n .

Given the output h(q), it is straightforward to compute the transformation T(q) associated with h(q). In the special case where $\dot{\xi}$ represents the operational space velocity error, then T(q) denotes the Jacobian matrix J(q).

According to [1],

$\ddot{q} = H^{-1}(q)[\tau - n(q, \dot{q})]$

with the nonlinear terms represented by

$n(q, \dot{q}) = B(q, \dot{q})\,\dot{q} + C(q)$

Then yields

$\ddot{\xi} = \ddot{r} + \dot{T}\dot{q} + T(q)\,H^{-1}(q)\,[\tau - n(q, \dot{q})]$

Define the control input function $u = \ddot{r} + \dot{T}\dot{q} + T(q)\,H^{-1}(q)\,[\tau - n(q, \dot{q})]$

Now we may define a state $y(t) \in R^{2p}$ by $y = (\xi\ \dot{\xi})$ and write the manipulator dynamics as

$$\dot{y} = \begin{pmatrix} 0 & I_p \\ 0 & 0 \end{pmatrix} y + \begin{pmatrix} 0 \\ I_p \end{pmatrix} u$$

This is a linear state-space system of the form $\dot{y} = Ay + Bu$ driven by the control input u.

EXERCISES

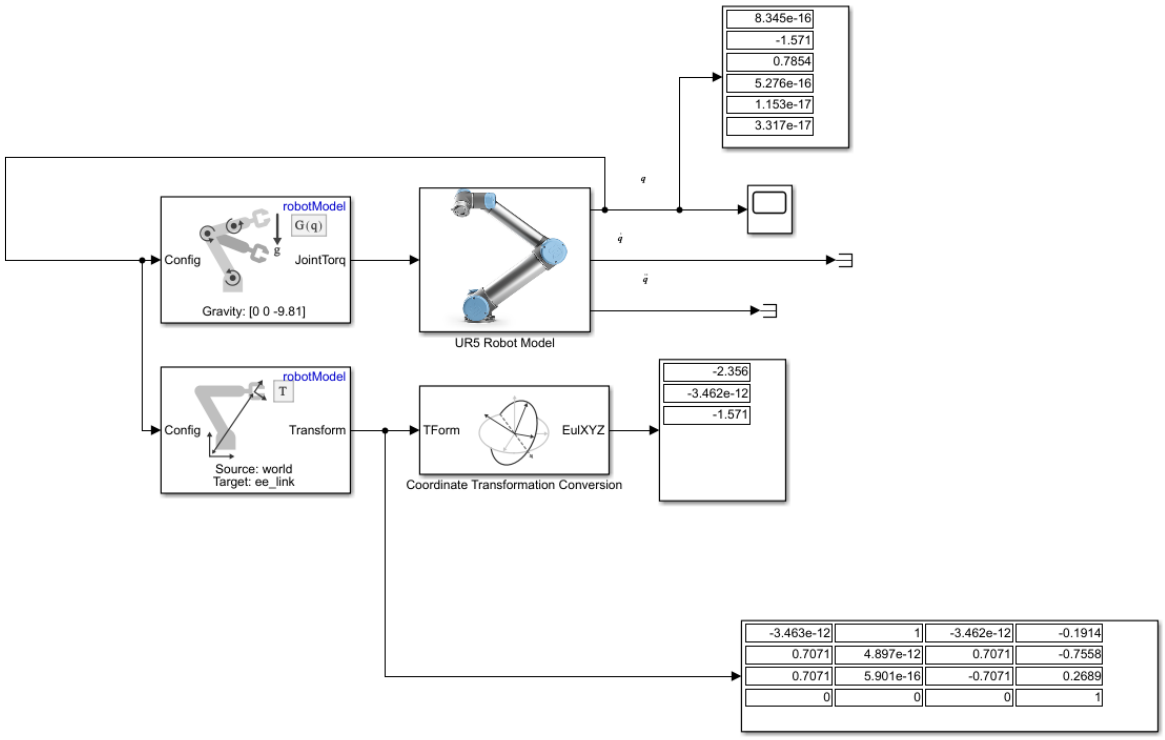
Most of the theory behind exercises has been explained before. So I'll not be redundant repeating everything in this section

1) Gravity

Here I used the simple formula for gravity compensation:

τ = C(q)– $K_v\ \dot{q}$

The bottom part has been needed for computation concerning the end effector space. So, its initial position and orientation



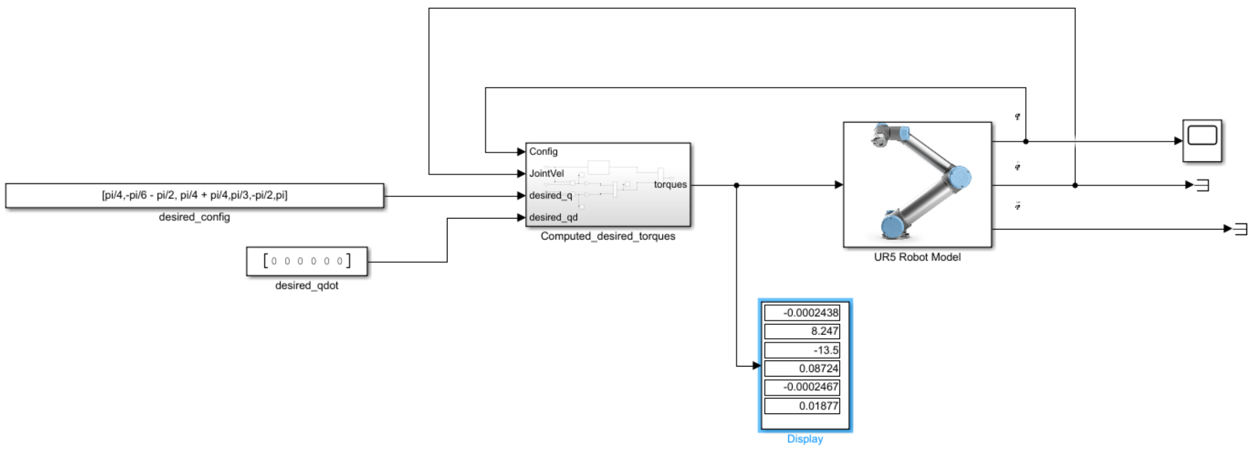
```
8.345e-16
-1.571
0.7854
5.276e-16
1.153e-17
3.317e-17
```

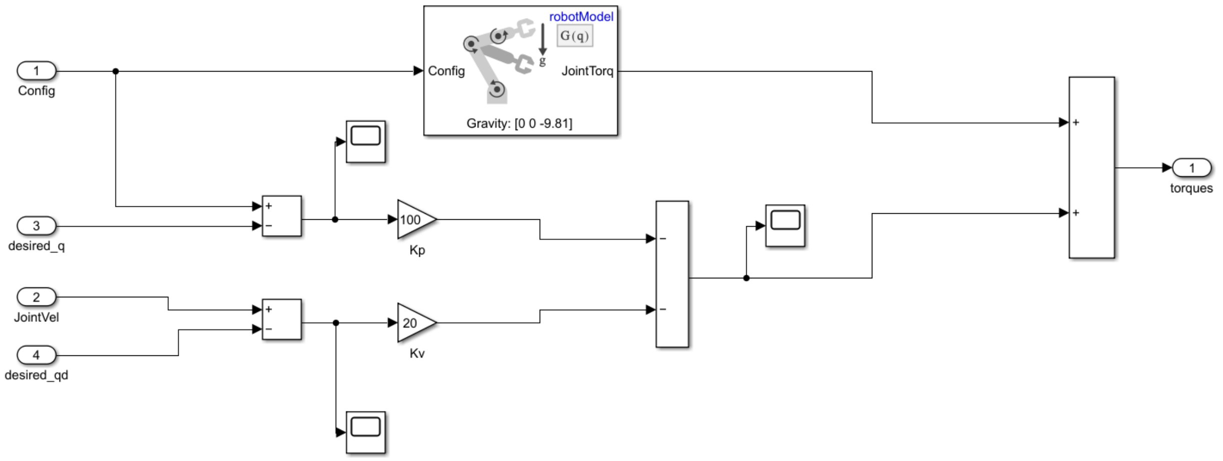
```
-2.356
-3.462e-12
-1.571
```

```
-3.463e-12          1   -3.462e-12    -0.1914
    0.7071  4.897e-12       0.7071    -0.7558
    0.7071  5.901e-16      -0.7071     0.2689
         0          0            0          1
```

2)Linear Joint Control

Given a desired joint configuration q* = q0 + Δq where q0 is the initial joint configuration and Δq = [pi/4, -pi/6, pi/4, pi/3, -pi/2, pi], provide torque commands in order to reach q*.



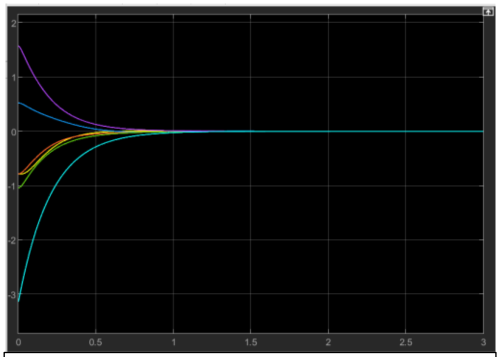


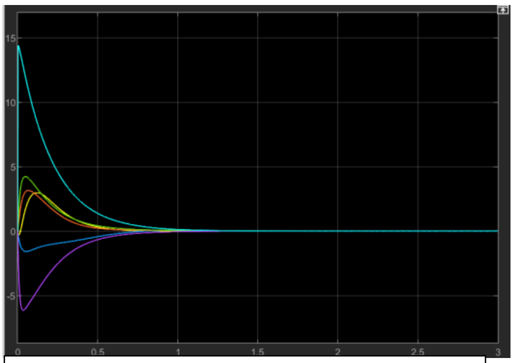
In this case I used the formula mentioned before:

τ = C(q)− $K_v\,\dot{q}$ - $K_p e_q$

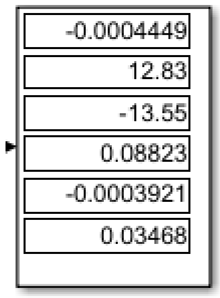
Where $e_q = q_d$ (t)−q(t)

We can see that our errors tends correctly to zero


Position error


Velocity error

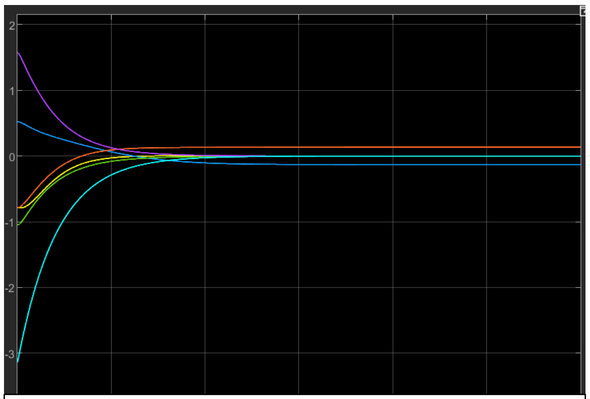
But what if Gravity compensation disappears?



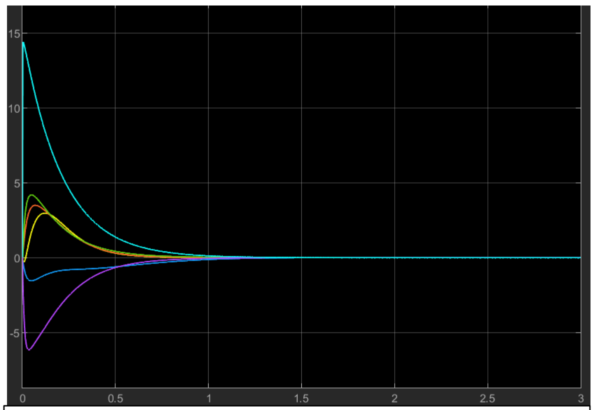
```
-0.0004449
12.83
-13.55
0.08823
-0.0003921
0.03468
```

If we delete the gravity compensation block we notice that the errors on the positions are not completely saturated:

The explanation is in the PD controller for regulation chapter



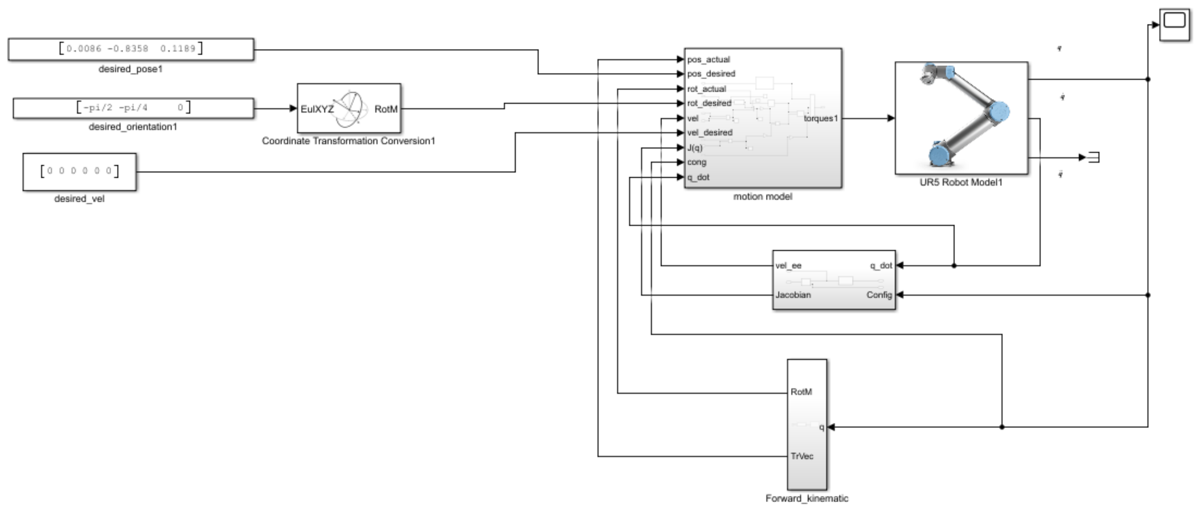
Position error

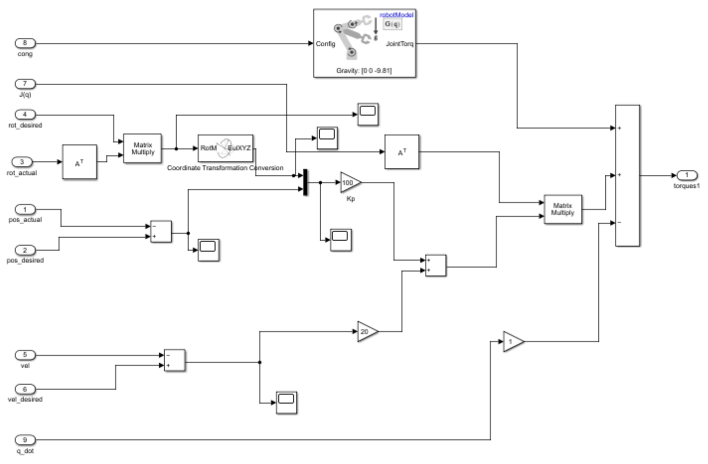


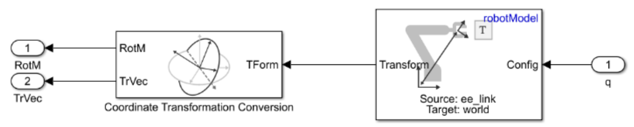
Velocity error

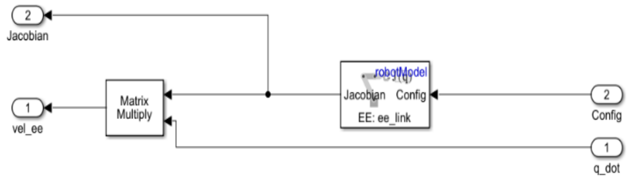
3)Linear Cartesian Control

Given a desired cartesian configuration x* = x0 + Δx where x0 is the initial cartesian pose of the end effector (‘ee_link’) and Δx = [0.2, -0.08, -0.15, pi/4, -pi/4, pi/2], provide torque commands in order to reach x*.









In this system, the joint positions, velocities, and accelerations are computed using standard rigid-body Robot Dynamics. The generalized force input Q is given by the PD Control law on the task-space error, scaled to the joint-space via a Jacobian-Transpose style control:

Recalling:

$$T = \begin{bmatrix} R & X \\ 0 & 1 \end{bmatrix}$$

$$\frac{d}{dt}\begin{bmatrix} q \\ \dot{q} \end{bmatrix} = f_{din}\,(q, \dot{q}, Q, F_{ext})$$

$$Q = J(q)^T\,(K_p\,E_T + K_d\,E_v) - B\dot{q} + G(q)$$

$$E_T = \begin{bmatrix} e_{rot} \\ e_{trans} \end{bmatrix} =$$

$$E_v = (v_{ref} - J(q)\,\dot{q})$$

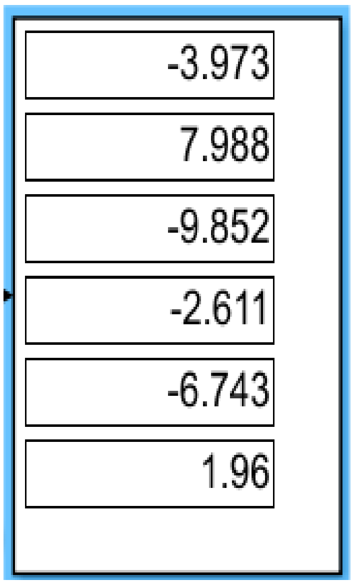
where:

- e_{rot} : is the rotational error converted to Euler angles using rotm2eul($R_{ref}\ {R_{act}}^{T}$)
- e_{trans} : is the error in *xyz*-coordinates, calculated as $X_{ref}-X_{act}$.
- G(q) : are the gravity torques and forces for all joints to maintain their positions in the specified gravity.
- J(q) : is the geometric Jacobian for the given joint configuration for more information, see the geometricJacobian function.

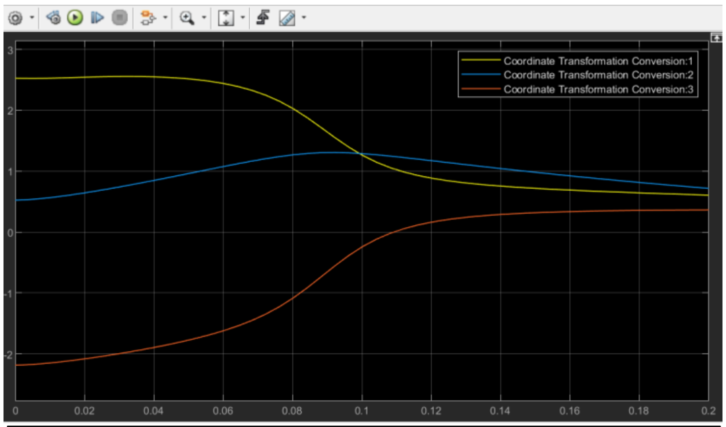
The control input relies on these user-defined parameters:

- K_p : Proportional gain, specified as a 6-by-6 matrix
- K_d : Derivative gain, specified as a 6-by-6 matrix
- B : Joint damping vector, specified as a two-element vector of damping constants in N·s·rad−1 for revolute joints and N·s·m−1 for prismatic joints

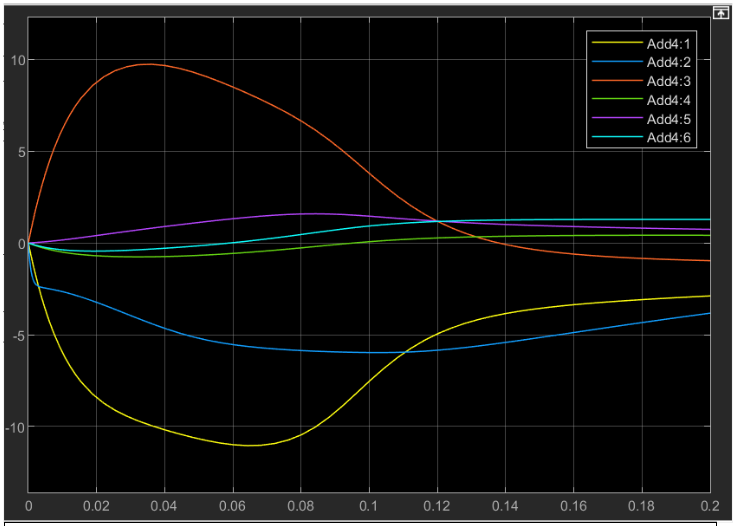
And here my results:



-3.973
7.988
-9.852
-2.611
-6.743
1.96



Rotation errors



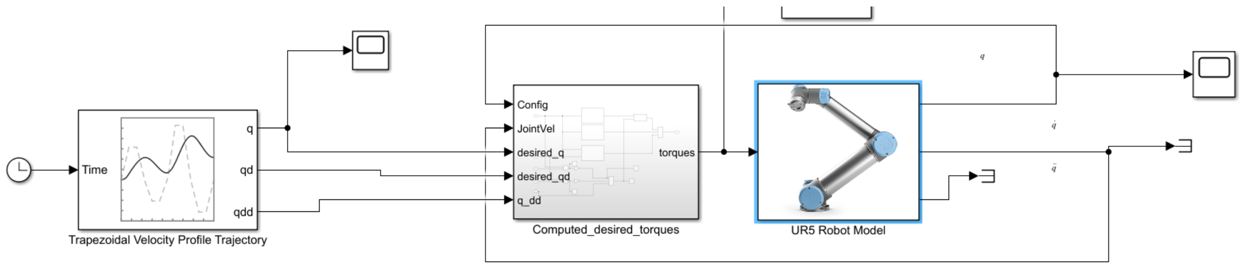
Velocity errors

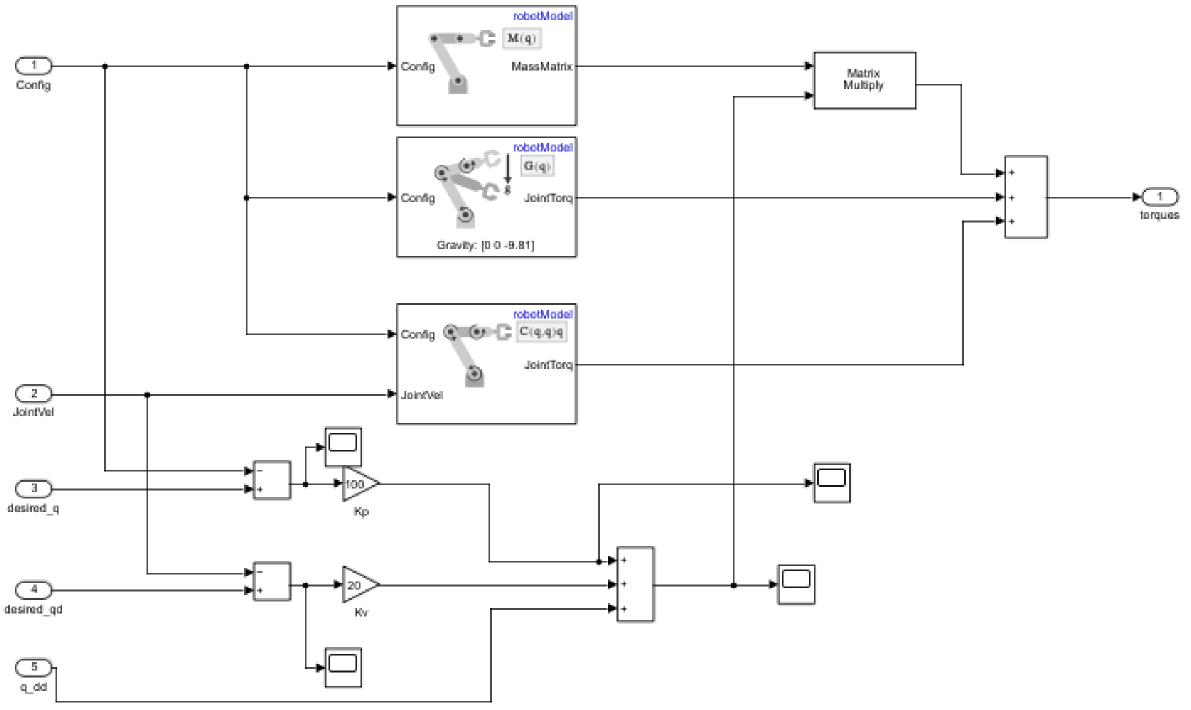
4)Computed Torque Control

In industrial robot control systems even a simple regulation from a setpoint to another (like in Ex.2 and Ex.3) is done via joint trajectory tracking. Consider again q* from Ex.2 and generate joint-space trajectories in order to reach it from q0. (There are blocks which do it for you, check the docs)

A) Use feedback linearization to implement a computed torque control in order to track the desired joint trajectories.

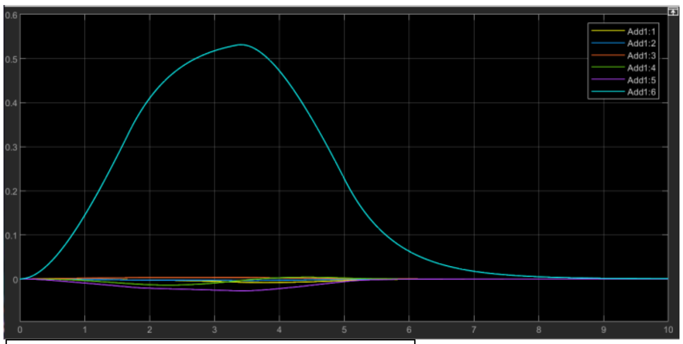
B) Open the robot model and check the joint objects: at the label 'Internal Mechanics' you will notice that a damping coefficient is provided to simulate joint friction. Try to remove it from each joint and compare the performance of the computed torque controller



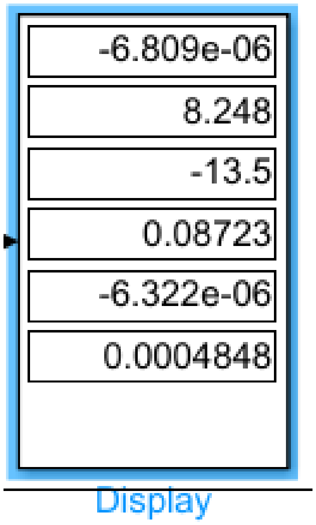


In this exercise I performed what I described in the section Inverses Dynamic Control in the tracking chapter.

And these are my results with dumping equal to 1

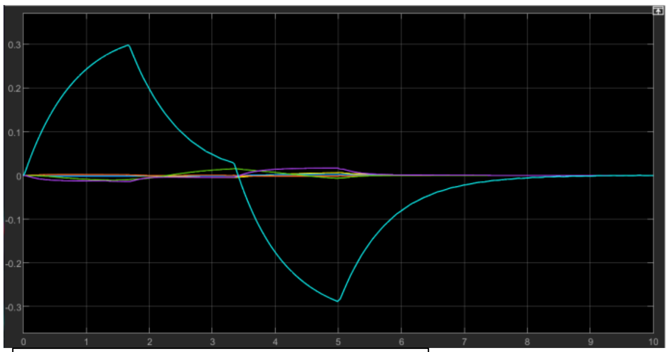


Position Error



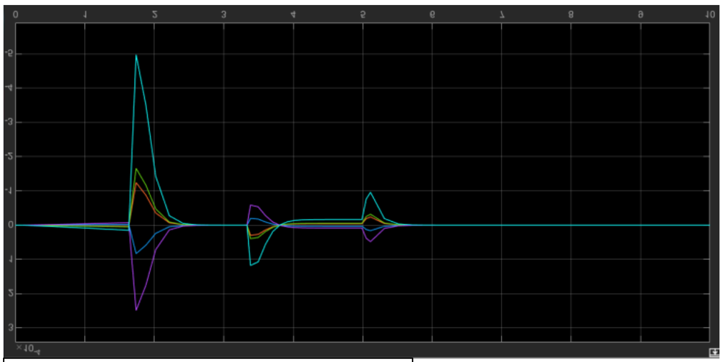
```
-6.809e-06
8.248
-13.5
0.08723
-6.322e-06
0.0004848
```

Display

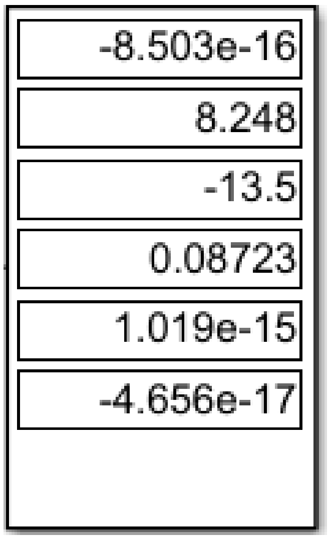


Velocity Error

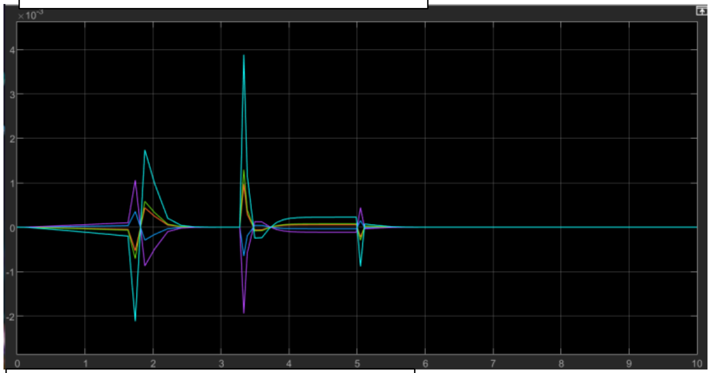
these are my results with dumping equal to 0



Position Error



```
-8.503e-16
8.248
-13.5
0.08723
1.019e-15
-4.656e-17
```



Velocity Error

If the dumping ratio is between 0 and 1, the system poles are complex conjugates and lie in the left-half s plane. The system is then called underdamped, and the transient response is oscillatory.

It came from $\delta\ddot{q} = -K_v\delta\dot{q} - K_p\delta q$:

We know that if all eigen values have real part less than zero the system is asymptotically stable

We can rewrite it in the following form (more familiar):

$$\ddot{y} + a\dot{y} + by = 0$$

$$\ddot{y} + 2\xi\omega_n\dot{y} + \omega_n^2 y = 0$$

Where

ξ is the dumping coefficient

ω_n is the frequency of the system

If I want to find the root of our system in order to understand its behaviour I'll write the system in the following form

$\lambda^2 + 2\xi\omega_n\lambda + \omega_n^2 = 0$

We want to find the roots of the second order polynomial

$\lambda = \frac{1}{2}\,[-2\xi\omega_n \pm 2\sqrt{4\xi^2\omega_n^2 - 4\omega_n^2}]$

$\lambda = -\xi\omega_n \pm \omega_n\sqrt{\xi^2 - 1}] = \omega_n[-\xi + \sqrt{\xi^2 - 1}]$

$\omega_n > 0\ always$

Note:

$K_p = \omega_n^2$ (And so the larger is ω_n and the larger is the stiffness of our robot)

For what concerns ξ:

- $\xi > 1$
  [ ]<0 and so we'll have two distinct eigenvalues λ_1, λ_2
- $\xi = 1$
  $\lambda_1 = \lambda_2 = -\omega_n$
- $0 < \xi < 1$

  Greater than zero because we want our dynamics being asymptotically stable

  $\lambda_{1/2} = -\omega_n[-\xi + i\sqrt{1 - \xi^2}]$

  So in this last case we'll have a oscillatory response and that's why that behaviour in the plots when dumping is put to 0