TÉCNICO LISBOA

Model predictive control for assistive robotic manipulation

Fábio Daniel Pires da Luz

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Bruno João Nogueira Guerreiro
Doutor Manuel Ricardo de Almeida Rodrigues Marques

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Bruno João Nogueira Guerreiro
Member of the Committee: Prof. José António Da Cruz Pinto Gaspar

November 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to express my gratitude to the incredible people that followed me through the 5 years studying at Instituto Superior Técnico.

Firstly, I deeply thank my supervisors Prof. Bruno Guerreiro and Doutor Manuel Marques for all the patience and confidence they gave me during the development of this dissertation. Without their availability, this process would be much more difficult and demanding than what already was.

To my family, especially to my parents, I would like to show my appreciation for providing me the best education I could ever had and for always being there for me, no matter what. To Bárbara, for the constant love and support, and for being a light during the rough times of the last few years, I could not thank you enough.

Last but not least, I would like to thank to my friends of ”boas”, Baeta, Bernardo, Coimbra, Guterres, Lopes, Miguel and Tom, for the friendship and for the help during all my academic path. I will always be there to share your victories and to carry you through the struggles of life, as I know you do to.

To all, thank you.

This work was partially funded by the FCT projects CAPTURE (PTDC/EEI-AUT/1732/2020) and REPLACE (LISBOA-01-0145-FEDER-032107), which include Lisboa 2020 and PIDDAC funds, and also project LARSYS (UIDB/EEA/50009/2020).

Abstract

The use of robots can be very beneficial for elderly and disabled people's assistance as they can help them regain autonomy to perform activities of daily living otherwise unachievable. In recent years, many studies have been made on Assistive Robotic Manipulators. However, a fully autonomous system that can adapt in real-time to the users' behavior and the surrounding environment is still yet to be accomplished.

To work towards this goal, in this thesis we aim to go beyond standard control techniques and use the Model Predictive Control (MPC) method for robotic manipulation, more specifically for a six-degrees of freedom robotic manipulator that is used for assistive purposes. MPC does not require any user's control input, allows the handle of nonlinear constraints, and opens the door to integrate machine learning and artificial intelligence techniques, contributing towards more autonomous assistance. The results show the ability to reach the reference position, even if it is variable through time, and to react immediately to perturbations, adapting the trajectory of the end-effector.

Keywords

Autonomous manipulators, Assistive technology, Model predictive control, Optimal control problem, Kinematic model

Resumo

A utilização de robôs pode ser muito benéfica para a assistência de idosos e pessoas com deficiência, uma vez que podem ajudá-los a recuperar a autonomia para realizar atividades da vida diária que de outra forma seriam inatingíveis. Nos últimos anos, muitos estudos foram feitos sobre manipuladores robóticos de assistência. No entanto, um sistema totalmente autónomo que consiga se adaptar em tempo real ao comportamento dos seus utilizadores e do ambiente circundante ainda está por ser realizado.

Com vista a trabalhar nesse sentido, nesta tese pretendemos ir além das técnicas de controlo padrão e utilizar técnicas de Controlo Preditivo Baseadas em Modelos (do inglês, MPC) para manipulação robótica, mais especificamente para um manipulador robótico de seis graus de liberdade usado para fins assistivos. O MPC não requer nenhum controlo por parte do utilizador, permite o lidar com restrições não lineares e abre a porta para a integração de técnicas de aprendizagem automática e inteligência artificial, contribuindo para uma assistência mais autónoma. Os resultados mostram a capacidade de atingir a posição de referência, mesmo que esta seja variável ao longo do tempo, e de reagir imediatamente às perturbações, adaptando a trajetória do efetor final.

Palavras Chave

Manipuladores autónomos, Tecnologia assistiva, Modelo de controlo preditivo, Problema de controlo ótimo, Modelo cinemático

Contents

List of Figures

Acronyms

ADL Activities of Daily Living

ARM Assistive Robotic Manipulator

DOF Degrees of Freedom

MPC Model Predictive Control

NLP Nonlinear Programming

NMPC Nonlinear Model Predictive Control

OCP Optimal Control Problem

PID Proportional, Integral, Derivative

RHC Receding Horizon Control

1

Introduction

Contents

1.1 Motivation

A robot is an automatically controlled, reprogrammable, multipurpose, mechanical system that may be either fixed in place or mobile [1]. During the last few years, the number of robots both at work in the industry and at the use of common people for Activities of Daily Living (ADL) have been increasing due to their decrease in price and increase in effectiveness [2]. In 2020, the International Federation of Robotics (IFR) reported a 12% increase in the global operational robot stock [3].

In this work, we focus on Assistive Robotic Manipulator (ARM), robotic manipulators designed to interact with humans and assist them in ADL in a safe manner. As defined in [4], an ARM is any robotic arm that is not body bound with a minimum of six-Degrees of Freedom (DOF), the required ones to model and simulate almost fully a human arm, and a gripper, to be able to hold objects, which is safe to interact with humans, especially individuals with special needs or disabilities, and can assist with several activities under the control of the user. Since these robots are specially designed for older people or individuals with upper-body disabilities, their utilization allows the users to regain some autonomy and take control of their lives by giving them the possibility to perform tasks otherwise unachievable without the help of another human, like feeding, drinking, dressing and, for example, reaching objects placed on tall shelves.

In fundamental human value activities' like eating a meal, the use of a joystick to control the manipulator [5] or the use of input buttons to select the type of food in the plate that is going to travel to a predefined mouth's position, as featured by the Handy 1 robot [6], are a few examples of ARMs that can be controlled by the users. However, these examples gather some concerns: it requires training to get familiarised with the system, and some users can not control the inputs due to their disabilities. So, to overcome these situations, the solution is to design fully autonomous robots, robots that can perceive the environment and act accordingly to it.

Conversely to industrial robots, ARMs usually do not have a strict environment to work upon. In the task of feeding, the capacity to perceive and adapt to changes in the pose of the head or the location of the mouth without losing the effectiveness to feed the user is extremely important. With the recent growth of the developments in assistive technologies, for example in the field of computer vision, such as the case of Feedbot [7], it is already possible to fully adapt to each individual's behavior.

One of the challenges to this adaptation is the design of the controllers used to fulfill the tasks attributed to the manipulators. Standard control techniques, like Proportional, Integral, Derivative (PID) control, can lack on stability and optimality and even on key requirements to fully describe a robotic system in order to make it fully autonomous, since it does not depend on the model of the ARM, has no knowledge of the system's constraints and can only handle one input and one output [8], unless it has an outer-loop that integrates all DOF of the robot [2]. In this dissertation, a Model Predictive Control (MPC) algorithm is proposed that uses a model of a manipulator and can plan a trajectory adapting in real-time

to changes in the users' mouth.

1.2 Objectives and contribution

Considering the previous section, the general objective of this dissertation project is to go beyond standard control techniques for robotic manipulation, like PID control. The goal is to design an algorithm based on the MPC method for a six-DOF robotic arm used for aid in ADL like assistive feeding purposes. The controller must be able to plan a trajectory and adapt in real-time to perturbations on its conditions, such as disturbances on the position of the end-effector and changes on its final reference position, the users' mouth.

To reach this primary goal, while reviewing the literature of previous research done about robotic manipulators, particularly the ones used in assistive feeding, control techniques and MPC problems, precise objectives were settled.

- Understand the robotic arm structure and configuration in to characterize its movement using homogeneous transformations and to solve the forward kinematics problem.
- Develop a dynamic model based on the robotic manipulator to simulate its behavior in the algorithm and determine the model's parameters that approximate the system response to reality.
- Study the MPC and its advantages over standard control techniques, design a cost function and implement a solution for the problem, having into consideration the system's constraints.

The designed MPC solution and algorithm are independent of the arm manipulator used and it is only necessary to replace the model and forward kinematics description with the correct one. In this thesis work, the robotic arm used was the six-DOF Niryo One [9] and as a consequence, the transformations calculated for the forward kinematics and the model were based on it. The code and the description on how to run the MPC algorithm can be found in this Github repository: https://github.com/fabio-luz/MPC_Arm.

1.3 Structure of the document

This thesis consists of 6 chapters. The present one, Chapter 1, is the introduction where the problem that the dissertation covers is described, as well as its main objectives, and where the outline of the document is stated.

Chapter 2 focuses on the state-of-the-art of the topics that are considered throughout this work. It presents a literature review on robotic manipulators, in particular on assistive robotic arms, and research on the control techniques that are used for those purposes. The main challenges regarding ADL such as feeding are analyzed and it is also presented research on the advancements of robotic manipulation using MPC.

Chapter 3 provides a description of the structure of the arm manipulator used in the project, as well as the coordinate system used, and the forward kinematics necessary to relate the state of the joints and the position of the end-effector in the world frame. The model used in simulation and the identification of its parameters are also discussed here.

Chapter 4 includes a brief theoretical background on the MPC method. It follows with an extensive analysis of the controller applied to the problem and model described in the previous chapters and its characteristics such as the cost function and constraints.

In Chapter 5, the results of the simulation work are presented, highlighting not only the state of the robot and the position of the robot during the trajectory but also the velocity and the control or the desired angle of each joint. Results are presented for both fixed and variable reference positions.

Chapter 6 draws the main conclusions of this thesis, and recommendations for future work are described.

2

Related work

Contents

This chapter covers the state of the art on the themes worked in this dissertation. A summary on the historical background of ARM is provided, and a review of the literature on the MPC method is presented, focusing on its application to robotic manipulation.

2.1 Assistive robotic manipulators

As defined in [10], an assistive robot provides aid and support to human users. This type of robots is aimed for elderly or disabled people, and its main objective is to contribute to their well-being, increasing their autonomy in daily life activities [11]. In the last decades, the global population is becoming older, an issue that affects all regions of the world, due to the increase of the average life expectancy and the decrease of the birthrate [12], which enlarges the importance of developing assistive technologies. There are several applications and research efforts on this subject, but the focus of this work is on assistive robotic manipulators aimed for physically disabled people, mainly with upper limb disabilities.

In recent years, many studies have been made on manipulators and their use to give assistance to humans in ADL like eating [6], [5], [13], [14], [7], dressing [15], and drinking [16]. One of the first low-cost robots to be developed and commercially available to aid severely disabled people was M. Topping's Handy 1 [6]. Created in 1987, the manipulator was mainly designed for the task of eating. It has a scanning system of lights that allows the user to select, using an input button, from seven different types of food from a specific plate. This Assistive Robotic Manipulator (ARM), even if very successful, was not autonomous as it could not be used by individuals that are not able to control the input buttons and used a predefined position for the users' mouth.

In [5], the authors asked the participants to eat bread from a plate, drink sips of soup from a bowl and open a door with a lever handle, using a six-DOF robotic manipulator and controlling the joints using a joystick. This study has the same issue as Handy 1 since it cannot be used by people who are incapable of maneuvering the joystick.

To surpass this problem, other robots have been developed in research efforts that operate using different types of user's control: in [13], the system has a man-machine interface controlled by the individual's head-motion, using a laser to point at the food; ICRAFT [14] offers an eye-tracking technology to control the manipulator, and a voice-controlled system is proposed in [15] for the task of dressing that enables the user to correct the trajectory of the end-effector of the arm.

Although these manipulators have more complex technology, they lack on adaptation to the user in real-time and still suffer from the fact that the control relies on user inputs that may not be possible to conceive for some individuals, and therefore they are not autonomous. A new kind of users' control was studied that used a Brain-Machine Interface (BMI) with Electroencephalography (EEG) signals to assist in the task of drinking [16]. In this system, a RGB-D camera was utilized to detect the environment and track the users' mouth movement. Not requiring any physical ability from the individual to control the robot is a step towards autonomy. However, this type of systems required training that can be complex to use and be distressing for every day use in ADL.

Feedbot [7] integrates a vision system that allows the robotic arm to be more autonomous. It adapts the trajectory in real-time to the users' movements by using Discriminative Optimization [17] for tracking the users' head, rather than having a pre-programmed mouth's position. It also does not require control inputs from the individual.

2.2 Robotic manipulation using Model Predictive Control

The MPC method, combined with vision or machine learning techniques, can contribute to a fully autonomous ARM by overcoming the limitations of the studies presented in the last section. As described in [18], MPC uses a dynamic model of a system to estimate its behavior and optimizes that estimation to determine the best control option.

By using MPC to control the manipulator, that unlike standard control techniques like PID can handle nonlinear constraints, it is possible to generate a trajectory for the end-effector that adapts in real-time to users' behavior and consider other features such as the search of food in the plate in the task of eating a meal.

In [19] and [20], the authors study the use of Nonlinear Model Predictive Control (NMPC) with different parameters in industrial robots for path-following. The word to be written is given as the trajectory to be tracked beforehand, and the results display precision and stability in the performance of both studies.

As mentioned, MPC can be coupled with deep learning and artificial intelligence techniques, which can be very beneficial for the aid in ADL for disabled people. For example, the conjunction of the two can be able to identify patterns in the individuals' behavior in order to change the trajectory of the ARM accordingly.

In [21], a strategy that combines robust MPC with deep neural networks is studied to achieve fast and safe tracking control for a robotic manipulator, satisfying constraints and stability requirements. Another study that implements the incorporation of vision, artificial intelligence, and MPC techniques for the purpose of aggressive autonomous driving is presented in [22]. It only uses a monocular camera placed in the car to acquire the images and perceive the environment. Those images are then analyzed using

deep convolutional neural networks to generate the cost functions to be used for real-time trajectory optimization by the MPC.

Despite the many research projects on the use of Model Predictive Control for robotic manipulation, to the best of our knowledge, no study has been made applied to a six-DOF manipulator for assistive purposes. In this work, we start by analyzing the structure of the robotic arm considered to compute its forward kinematics required to connect the state of the joints to the position of the end-effector. The model that will be used in the algorithm is then discussed, and at last, the MPC method applied in the context of this problem is formulated and presented.

3

Six-DOF robotic arm

Contents

3.1 Structure and coordinate system

The manipulator used in this work is the Niryo One robot arm. It is considered a serial manipulator or a kinematic chain since its design is composed of several links or rigid bodies connected by joints. All the figures and mechanical specifications presented in this chapter were retrieved from the Niryo official website [9].

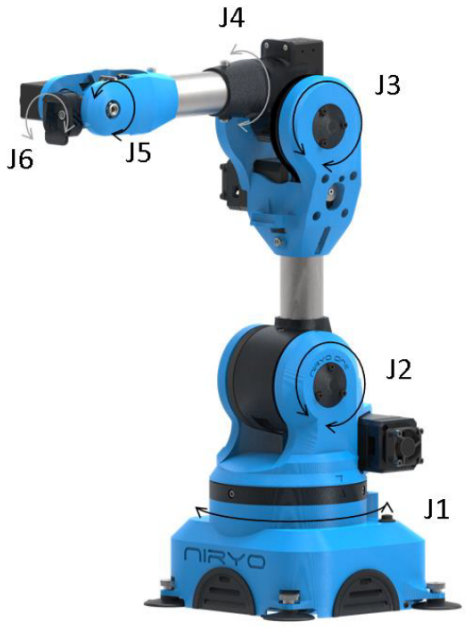


Figure 3.1: Niryo One robotic arm

As presented in Figure 3.1, Niryo One is a six-DOF robotic arm, meaning that it has six independent position variables [2]. Each link can only move in one direction, and so each DOF corresponds to a joint. It has three twisting joints and three rotational joints: twisting joints are the ones where the axis of rotation is parallel to the axis of the two links, rotating in a twisting motion (J1, J4, and J6), while rotational joints are the ones where the two links rotate relative to each other about a fixed axis, perpendicular to the links (J2, J3, and J5). In Table 3.1, the physical limitations of each joint are stated.

Joint	Min (rad)	Max (rad)
J1	-3.054	3.054
J2	-1.5707	0.628319
J3	-1.4101	0.994838
J4	-2.61799	2.61799
J5	-2.26893	2.26893
J6	-2.57	2.57

Table 3.1: Limits of each joint

In the center of each joint is defined a coordinate frame to be possible to compute the position of the end-effector relative to the base frame from the angles of the joints. This geometrical problem is called forward or direct kinematics, and it will be addressed in the next section.

Although there are many conventions to select the coordinate frames, for example, the Denavit-Hartenberg convention [23], for the sake of simplicity, the frame system for each joint was chosen in a way that every referential has the same orientation for the considered initial position of the manipulator, Figure 3.2, with the x-axis pointing in the direction of the reader, the y-axis pointing to the right, and with the z-axis pointing up.

In Figure 3.2 the physical dimensions of the manipulator are described - $L_1 = 103mm$, $L_2 = 80mm$, $L_3 = 210mm$, $L_4 = 30mm$, $L_5 = 41.5mm$, $L_6 = 180mm$, $L_7 = 23.7mm$ and $L_8 = 5.5mm$. The base referential frame, positioned in the center of the base of the robotic arm, and the coordinate frames of each joint chosen are also represented in red.

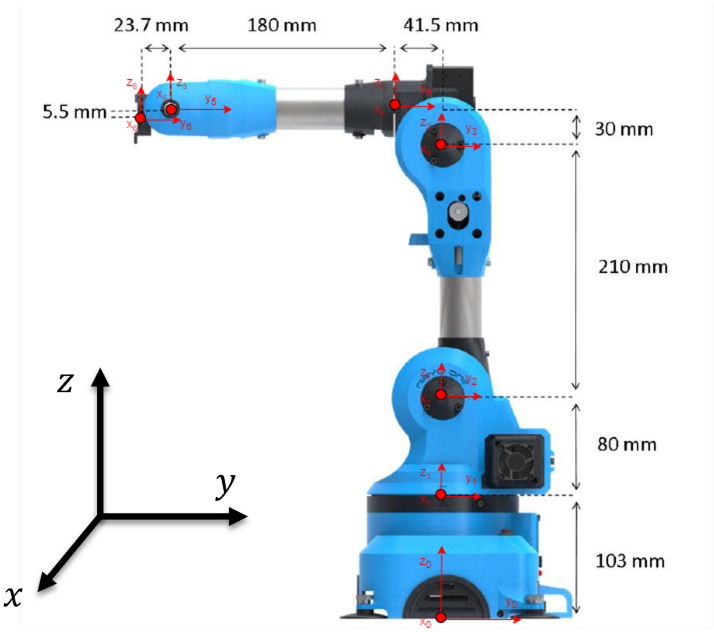


Figure 3.2: Dimensions of the links and coordinate frames

3.2 Forward kinematics description

The direct kinematics consists in finding the position and orientation of the end-effector, which is located in J6, relative to the base referential defined in the base of the serial manipulator, given the value of the angles of each joint, θ_i for $i = 1, ..., N = 6$.

Knowing the angle of the joint between two consecutive links, it is possible to compute the homogeneous transformation between them. Let R_i^j be the rotation matrix that transforms a vector in the i-th coordinate frame to a coordinate frame which is parallel to the j-th coordinate frame and p_i^j be the position of the i-th coordinate frame with respect to the j-th coordinate frame. Having in regard the coordinate system defined in Figure 3.2, the following transformations can be observed.

- Base referential $\rightarrow$ J1: rotation around the z-axis, translation of L_1 in the z-axis.
- J1 $\rightarrow$ J2: rotation around the x-axis, translation of L_2 in the z-axis.
- J2 $\rightarrow$ J3: rotation around the x-axis, translation of L_3 in the z-axis.
- J3 $\rightarrow$ J4: rotation around the y-axis, translation of $-L_5$ in the y-axis and L_4 in the z-axis.
- J4 $\rightarrow$ J5: rotation around the x-axis, translation of $-L_6$ in the y-axis.
- J5 $\rightarrow$ J6 (end-effector): rotation around the y-axis, translation of $-L_7$ in the y-axis and $-L_8$ in the z-axis.

All the rotation matrices and translation between the coordinate referentials above are defined as

$$
\begin{aligned}
R_1^0 &= \begin{bmatrix} c_1 & -s_1 & 0 \\ s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & p_1^0 &= \begin{bmatrix} 0 \\ 0 \\ L_1 \end{bmatrix} & R_4^3 &= \begin{bmatrix} c_4 & 0 & s_4 \\ 0 & 1 & 0 \\ -s_4 & 0 & c_4 \end{bmatrix} & p_4^3 &= \begin{bmatrix} 0 \\ -L_5 \\ L_4 \end{bmatrix} \\
R_2^1 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_2 & -s_2 \\ 0 & s_2 & c_2 \end{bmatrix} & p_2^1 &= \begin{bmatrix} 0 \\ 0 \\ L_2 \end{bmatrix} & R_5^4 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_5 & -s_5 \\ 0 & s_5 & c_5 \end{bmatrix} & p_5^4 &= \begin{bmatrix} 0 \\ -L_6 \\ 0 \end{bmatrix} \\
R_3^2 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_3 & -s_3 \\ 0 & s_3 & c_3 \end{bmatrix} & p_3^2 &= \begin{bmatrix} 0 \\ 0 \\ L_3 \end{bmatrix} & R_6^5 &= \begin{bmatrix} c_6 & 0 & s_6 \\ 0 & 1 & 0 \\ -s_6 & 0 & c_6 \end{bmatrix} & p_6^5 &= \begin{bmatrix} 0 \\ -L_7 \\ -L_8 \end{bmatrix}
\end{aligned}
\tag{3.1}
$$

where c_i and s_i correspond, respectively, to $\cos(\theta_i)$ and $\sin(\theta_i)$.

The transformation matrix T_i^j is a concatenation of the 3-by-3 rotation matrix with the 3-by-1 position array, both between two consecutive coordinate frames and describes the frame i relative to the frame j.

$$
T_i^j = \begin{bmatrix} R_i^j & p_i^j \\ 0_{3\times1} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}
$$

For example, the transformation matrix between the base referential and the frame in J1 is given by

$$T_1^0 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

With all the transformation matrices defined between the consecutive frames, it is possible to compute the relation between the base referential frame and the coordinate frame of J6 where the end-effector is located, as the product of all the consecutive transformations, defined as

$$T_6^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 = \begin{bmatrix} R_6^0 & p_6^0 \\ 0_{3\times1} & 1 \end{bmatrix} \tag{3.4}$$

noting that the position of the end-effector in the world frame is then expressed in the last column of the transformation, by p_6^0.

3.3 Model dynamics

In the previous section, it was only taken into consideration the kinematic aspect of the manipulator, the motion without regard to the forces which cause it [2]. A dynamic model is required to simulate with accuracy the behavior of the robotic arm and contemplates not only the torques applied by the motor actuators present in each joint but also the external forces suffered by the manipulator. The dynamic model of a serial manipulator can be computed using the Newton-Euler [24], or Lagrangian methods [25], and its general expression is described in [26] by

$$\tau = M(x)\ddot{x} + C(x, \dot{x}) + G(x) \tag{3.5}$$

where τ represents the actuators' torques while x, $\dot{x}$ and $\ddot{x}$ are the position, velocity, and acceleration of the joints, respectively. $M(x)$ denotes the inertia or mass matrix, $C(x, \dot{x})$ is the vector that includes the Coriolis and centrifugal forces, and $G(x)$ is the gravity vector.

Given that the maker of this arm does not provide the necessary information, the procedure to identify the center of mass and moment of inertia of each link is a complex task, which requires disassembling the manipulator and using specialized measuring apparatus. On the other hand, the type of actuators used in each joint (stepper motors) takes as a command a desired angular position and not a current or torque.

Consequently, the torques of the actuators could not be computed, and a much simpler and general differential system model was defined and used in the simulation that only considers the angular position's dynamics of the joints and admits a control input to be a desired angular position. The model is thus described as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= p_6^0(x(t)) \end{aligned} \tag{3.6}$$

where x is the vector of position/angle of each joint, meaning $x = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$, and u is the control signal, the desired angle for each joint, $u = (\theta_{1,des}, \theta_{2,des}, \theta_{3,des}, \theta_{4,des}, \theta_{5,des}, \theta_{6,des})$. A and B are two diagonal matrices that come from the identification of the system's parameters, $A = diag(a_1, a_2, a_3, a_4, a_5, a_6)$ and $B = diag(b_1, b_2, b_3, b_4, b_5, b_6)$.

As stated in Equation (3.6), the output of the system is defined by the position of the end-effector, located on J6, in the base referential frame, p_6^0. It is computed by the forward kinematics described in the previous section of this document.

3.3.1 Parameters' identification

The transfer function for a general system like the one used for the model described in Equation (3.6) for a single joint is given by

$$\begin{aligned} & X_i(s)(\tau_i s + 1) = U_i(s)K_{0i} \quad \Leftrightarrow \\ \Leftrightarrow \quad & \frac{X_i(s)}{U_i(s)} = \frac{K_{0i}}{\tau_i s + 1} \end{aligned} \tag{3.7}$$

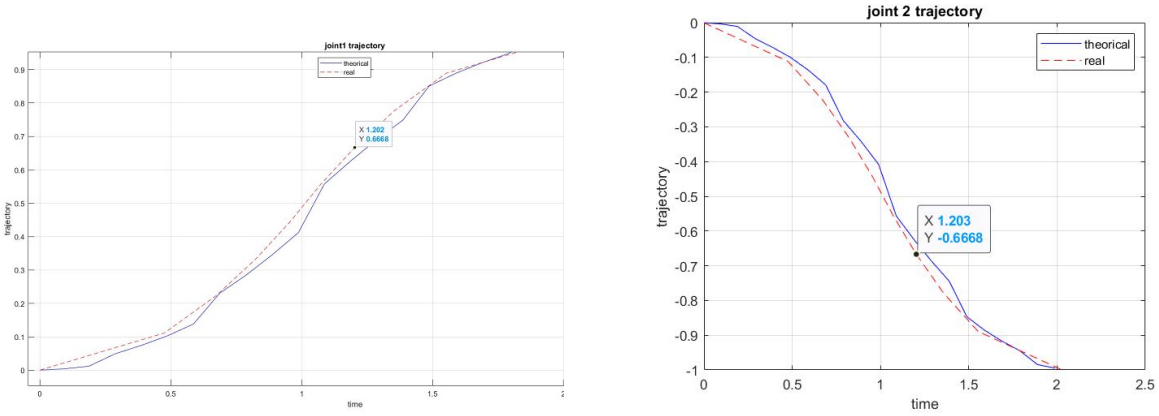
K_{0i} represents the static gain of a single joint while τ_i represents its time constant. The differential equation that corresponds to the presented transfer function is described as

$$\begin{aligned} & \tau_i \dot{x_i}(t) + x_i(t) = K_{0i}u_i(t) \quad \Leftrightarrow \\ \Leftrightarrow \quad & \dot{x_i}(t) = -\frac{1}{\tau_i}x_i(t) + \frac{K_{0i}}{\tau_i}u_i(t) \end{aligned} \tag{3.8}$$

Comparing Equation (3.8) to Equation (3.6), it is straightforward to correspond the model's parameters used in the diagonal matrices to the ones from the general differential system: $a_i = -\frac{1}{\tau_i}$ and $b_i = \frac{K_0 i}{\tau_i}$. Given the fact that the static gain is the gain between the steady-state of the output and the input and considering that the joints of the robotic arm always reach the same angle provided as an input, it will be assumed that $K_0 = 1$. By analyzing the response to a step input, it is possible to easily identify the time constant of the system and, consequently, the parameters of the model used in the simulation.

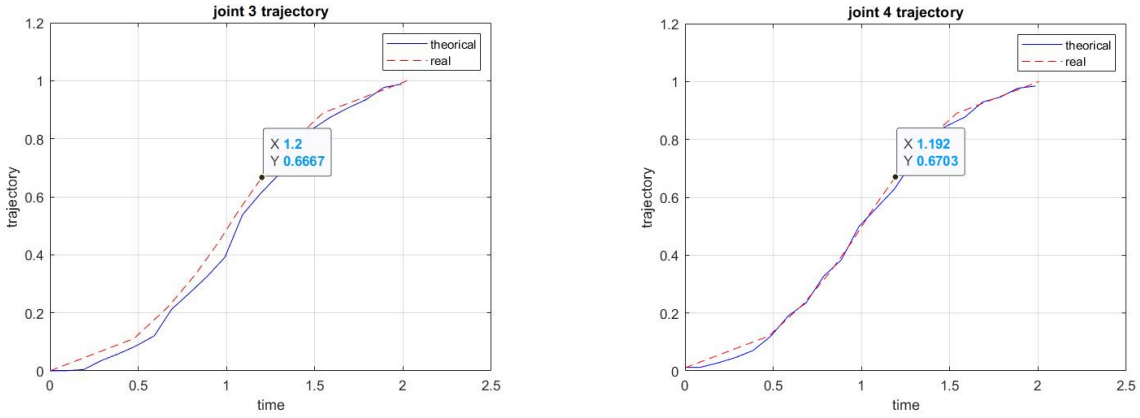
The Niryo One's *Matlab* interface [27] was used to send simple commands to the manipulator's joints to analyze its response in real-time. When a set of angles is sent to control the joints, a trajectory is

created and passed to the interface's controller that tries to execute it. Making the assumption that the joints' control signal sent to the arm to create the trajectory is a step input of the desired angle for the given joint, as in Equation (3.6), the time constant, τ, can be estimated by analyzing the graphical time response of the system since it is known that it represents the time that it takes for the step response to reach 63.2% of its final value. The following figures were taken directly from the Niryo One's *Matlab* interface.



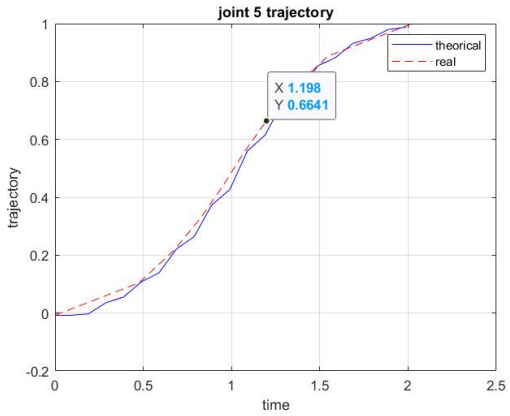
(a) Joint 1 **(b)** Joint 2

Figure 3.3: J1 and J2 response for 1 and -1

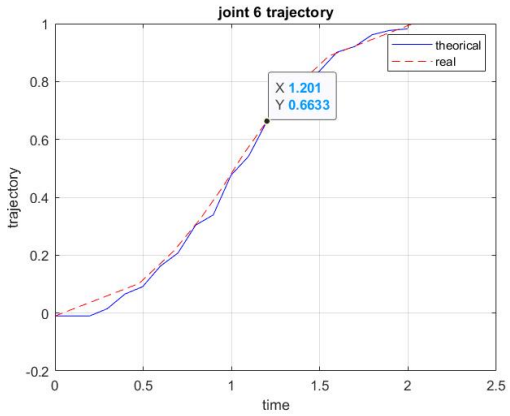


(a) Joint 3 **(b)** Joint 4

Figure 3.4: J3 and J4 response to the same angle - 1 radian



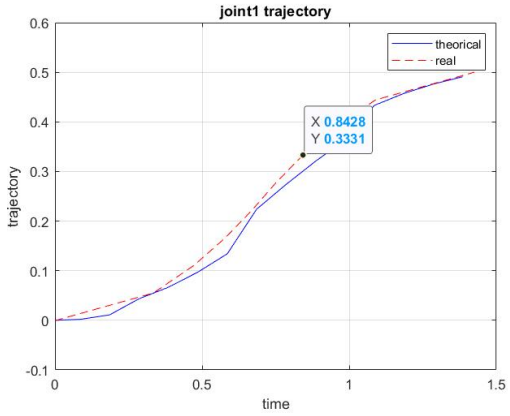
(a) Joint 5



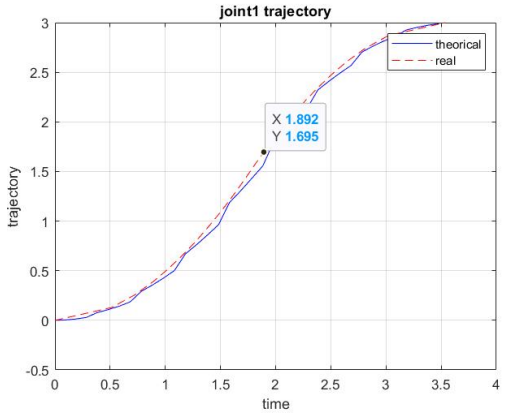
(b) Joint 6

Figure 3.5: J5 and J6 responses to the same angle - 1 radian

Between Figure 3.3 and Figure 3.5, responses are drawn for the six joints having as input the same angle, 1 radian, except for J2 where the input is -1 radian. In blue, it is represented the theoretical trajectory, while in red, it is represented the executed one.



(a) 0.5 radians



(b) 3 radians

Figure 3.6: J1 response to different angle inputs

In Figure 3.6 two examples for the same joint, having as input different angles, are represented. The color scheme is maintained in these figures.

Several different commands were sent to all the joints using the interface. Some conclusions could be drawn based on the responses analyzed, and the parameters of the model could be estimated.

- All joints have a similar response to the same command, reaching the angle at a similar time.
- Depending on the angle command, the response varies between 1.2 seconds and 3.5 seconds.
- For the deduced unit step input, an angle of 1 radian, all joints reach 63.2% of the final value in approximately 1.2 seconds. So, it is assumed that $\tau_i = 1.2$ for all the joints and, consequently, the parameters used in the simulation are $a_i = -0.83$ and $b_i = 0.83$ for all joints, which guarantees that the MPC converges in the time interval stated in the previous item.

Since it is an ongoing development effort to be able to send continuous control inputs to the joints of the robotic arm and thus analyze their response independently, it was not possible to explore more advanced identification techniques, like *ARMAX*, to identify the system's parameters.

4

Model Predictive Control

Contents

4.1 Theoretical background

Model Predictive Control (MPC) is a feedback control algorithm that uses an explicit dynamic model in order to make predictions about the future behavior of a given system in response to the actuation of the manipulated variables (input control). The dynamic state-space model of a system, described in continuous-time, can be defined by

$$\dot{x}(t) = f(x(t), u(t)) \tag{4.1}$$

where x is the state of the system, and u is the control function. If the model can be described as $\dot{x}(t) = Ax(t) + Bu(t)$ then it is considered a linear model, where A is the state matrix and B the input matrix.

Based on the state predictions and having in consideration the reference state or the desired trajectory, an Optimal Control Problem (OCP) is solved to determine the optimal control sequence to reach a given goal, subject to operating constraints [28]. The generic optimization problem solved by MPC to determine that sequence, given an initial state, x_0, is defined in continuous-time as

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & m(x(T)) + \int_0^T l(x(\tau), u(\tau))d\tau \\
\text{subject to:} \quad & x(0) = x_0 \\
& \dot{x} = f(x(t), u(t)) && t \in [0, T] \\
& \underline{h} \leq h(u(t)) \leq \overline{h} && t \in [0, T] \\
& \underline{x} \leq x(t) \leq \overline{x} && t \in [0, T] \\
& \underline{x}^e \leq x(T) \leq \overline{x}^e
\end{aligned} \tag{4.2}$$

As mentioned, at each sampling time, MPC uses the dynamic model, expressed in Equation (4.1) and in the second constraint of Equation (4.2), to simulate the response to several changes in the control signal in the following T iterations or time steps. T is called the prediction horizon and represents the length of time that is considered in each optimization problem.

The MPC simulates multiple actuation scenarios in the manipulated variables and the respective model response and chooses the control sequence that not only minimizes the defined cost function that usually includes the difference between the desired trajectory and the predicted one and the variation in input control signal between consecutive iterations, but that also respects the constraints. The cost function is given by the sum of the Lagrangian cost term or the stage cost, $l(x(t), u(t))$, and the Mayer cost term or the terminal cost, $m(x(T))$.

One of the advantages of using the MPC is that it can handle constraints in the state-space of the system during the path, represented by $\underline{x}$ and $\overline{x}$, at end-time, $\underline{x}^e$ and $\overline{x}^e$, and in the manipulated

control variables, $\underline{h}$ and $\overline{h}$, which is very helpful since it can take into account characteristics of real-world systems, such as physical limitations and desired velocities while predicting the future states and selecting the optimal trajectory to a given reference. One general example of the MPC approach is shown in Figure 4.1.

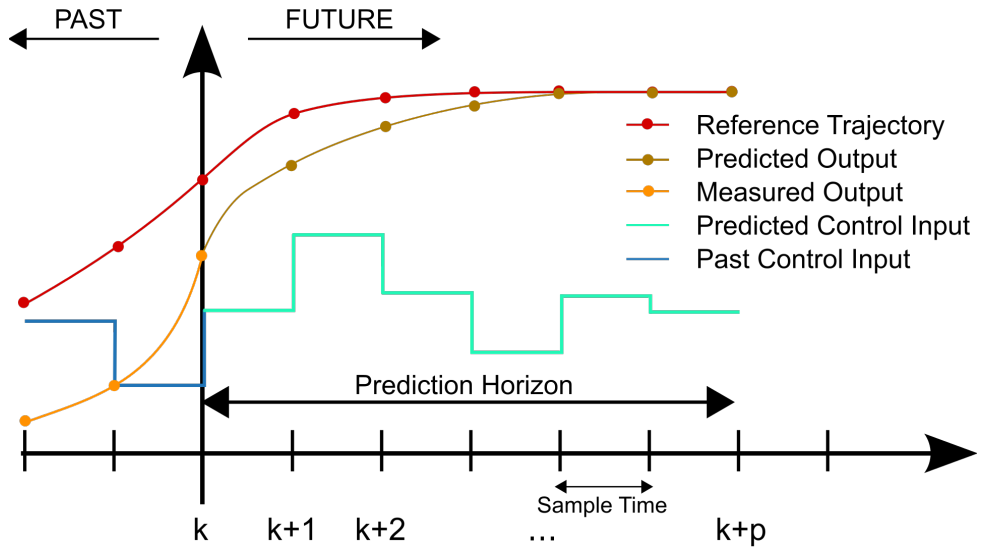


Figure 4.1: MPC discrete approach. Image taken from Wikipedia.[1]

After the optimal control sequence is determined, its first value denominated by optimal control action is applied in the system for the current sample, and thus a new state can be obtained, either through measurements for a real system, or using discretization methods for simulating the system, such as the forward Euler discretization method, described as

$$x(k+1) = x(k) + \Delta t\, f(x(k), u(k)) \tag{4.3}$$

where Δt is the sampling time and f is computed according to the dynamic model defined in Equation (4.1).

This new state can be different than the one predicted in the previous iteration of the controller due to non-considered real-world parameters in the simulation, like external forces. Then, the prediction horizon moves forward one time step, and the process is repeated for the new state of the system. The optimization is repeated at each sampling time based on the updated information from the system's state [28]. Given the fact that the prediction horizon moves forward in each iteration, the MPC can be also referred to as Receding Horizon Control (RHC).

[1] URL: https://commons.wikimedia.org/wiki/File:MPC_scheme_basic.svg

4.2 MPC applied to a six-DOF robotic arm

The simulation algorithm was developed using the *MATLAB* interface of *ACADOS*, a software package for the efficient solution of optimal control and estimation problems [29] that also makes use of *CasADi*, an open-source tool for nonlinear optimization and algorithmic differentiation [30], to define variables. The OCP that *ACADOS* can handle and the way that it is formulated can be found in the document [31].

The simulation algorithm diagram is represented in Figure 4.2. A link that contains the files with the code used in the simulation and an explanation of how to run the algorithm is presented in Section 1.2.

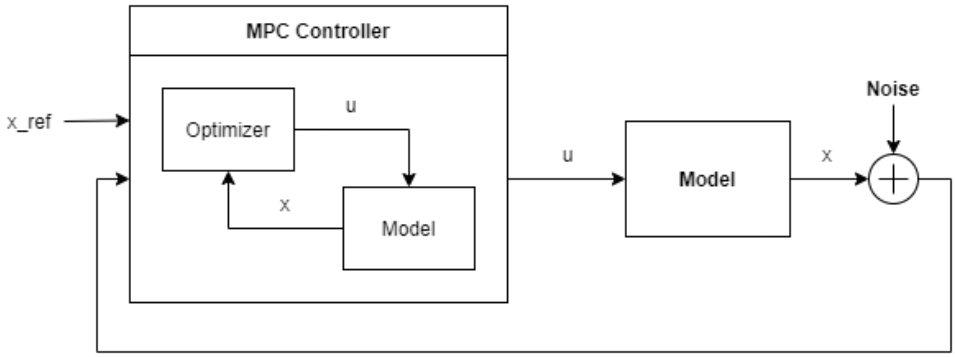


Figure 4.2: Simulation diagram

The model used both in the MPC controller to solve the OCP and to simulate the robot is defined in the previous chapter of this work.

The state used in this problem, x, is defined by the position/angle of the joints in radians, while the control, u, is given by the desired angle for each joint, as already mentioned. To initialize the MPC, the initial state of the joints, $\overline{x}_0$, and the reference position at end-time, y^e_{ref}, are generated according to the purpose of the simulation (fixed or variable reference position, physical limitations).

The general formulation of the OCP solved by *ACADOS* can be found in the problem formulation document [31] in Equation (1). The OCP applied to the problem developed in this dissertation is described as

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & m(x(T)) + \int_0^T l(x(\tau),u(\tau))d\tau \\
\text{subject to:} \quad & x(0) = \overline{x}_0 \\
& \dot{x}(t) = Ax(t) + Bu(t) \qquad t \in [0,T] \\
& \underline{h} \leq u(t) \leq \overline{h} \qquad t \in [0,T] \\
& \underline{x} \leq x(t) \leq \overline{x} \qquad t \in [0,T] \\
& \underline{x}^e \leq x(T) \leq \overline{x}^e
\end{aligned} \tag{4.4}$$

The dynamics, the second constraint on the OCP, are formulated as explicit equations in continuous time, as written in Equation (3.6). *ACADOS* offers different options to formulate both the dynamics and the cost function.

In addition to the first constraint, the initial state, $\overline{x}_0$, that can be viewed as the "current" position of the joints and that changes in every sample of the algorithm according to the optimal control option found, updating the OCP that is going to be solved in the next iteration, it is necessary to have in consideration the physical limitations of the arm both in the state and in the input since the control is the desired position of the joints. Those limitations were already stated in the previous chapter of this work in Table 3.1.

$$\begin{aligned} \underline{h} = \underline{x} = \underline{x}^e &= \begin{bmatrix} -3.054 & -1.5707 & -1.4101 & -2.61799 & -2.26893 & -2.57 \end{bmatrix} \\ \overline{h} = \overline{x} = \overline{x}^e &= \begin{bmatrix} 3.054 & 0.628319 & 0.994838 & 2.61799 & 2.26893 & 2.57 \end{bmatrix} \end{aligned} \tag{4.5}$$

In the last three constraints, h represents the nonlinear constraint of the control, while $\underline{x}$ and $\underline{x}^e$ are the lower bounds of the state constraint in the path and at end-time, respectively, and $\overline{x}$ and $\overline{x}^e$ are the upper limits. J_{bx} and J_{bx}^e are 6-by-6 identity matrices.

For this problem, the cost module used is the nonlinear least squares since it is more useful to define y and y^e as *CasADi* expressions.

$$\begin{aligned} l(x,u) &= \frac{1}{2}\|y(x,u) - y_{ref}\|_W^2 \\ m(x) &= \frac{1}{2}\|y^e(x) - y_{ref}^e\|_{W^e}^2 \end{aligned} \tag{4.6}$$

where W and W^e represent weight matrices that can be tuned, and $y(x,u)$ and $y^e(x)$ are defined as

$$\begin{aligned} y(x,u) &= [p_6^0(x), \dot{x}(x,u)] \\ y^e(x) &= p_6^0(x) \end{aligned} \tag{4.7}$$

The Lagrangian term not only has into consideration the position of the end-effector of the robotic arm, $p_6^0(x)$, but also the velocity of the joints, $\dot{x}(x,u)$, to improve stability and decrease rash movements. The Mayer term is only added at the end of the optimization process, and it can not depend on the control signal, u, so it is only defined by the position of the end-effector.

The end-effector is located in the last joint of the arm, J6, and its position can easily be computed knowing the angle of the joints, knowing the state, as demonstrated in Section 3.2. On the other hand, since the state and control (the desired angles) of the joints are known at each sampling time, the velocity at a given time can be computed using the model's dynamics stated in Equation (3.6).

As mentioned, the reference position at end-time, y_{ref}^e, is an input of the algorithm and is assumed that it is valid and reachable within the physical limitations of the arm, according to the forward kinematics coordinate scheme used. If the given reference position is not reachable, the MPC algorithm does

not converge and tends to the last reachable position in the trajectory. The reference during the path trajectory, y_{ref}, is given not only by reference position but also by the reference velocity for each joint, which should be zero in order to minimize it.

Regarding the other parameters of the optimization problem presented in Equation (4.4), it was defined a prediction horizon, T, of 0.5 seconds and 10 shooting nodes or samples, N, which corresponds to a sampling time, Δt, of 0.05 seconds.

The arm model is then translated to the model used by the OCP, along with the initial state, output reference, dynamics, constraints, and cost modules. Also, other options offered by *ACADOS* are defined, such as the simulation method (explicit Runge Kunta integrator that had been chosen to write the dynamics explicitly) and the Nonlinear Programming (NLP) solver (sequential quadratic programming). All the definitions available are stated in *ACADOS* problem formulation document [31].

Once one OCP is solved, and an optimal control action to reach the desired position is found, the next iteration state is determined. First, the velocities of the joints, $\dot{x}$, are calculated according to the model defined in Equation (3.6). Then, the new state is computed using the sample time, the previous state and the control, using the forward Euler method, described in Equation (4.3).

To do an approximation to reality where the arm is subjected to external forces and not always follows the trajectory without any errors, random noise is added to the position of each joint if the end-effector is not within a radius of 1 from the reference position. The noise ranges from -0.05 and 0.05, and this value was defined based on how much the angle of the joints changes from a sample to the next one. In addition, if the new state with the added noise does not respect the physical limits of the arm in one or more joints, the correspondent noise is discarded, and a new random value is added. When the new state is determined, it is defined as the initial state of the next OCP, beginning a new iteration in the MPC module.

5

Simulation results

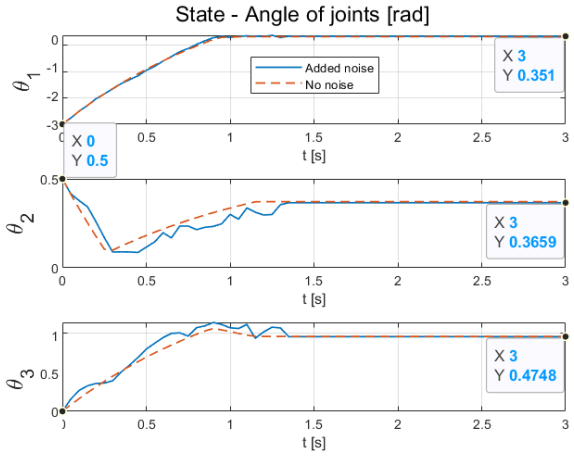
Contents

In this chapter, the simulation results are presented. Three scenarios are shown: (a) a pair of an initial state and a final reference position for the end-effector of the robotic arm, (b) two different reference positions in the same trajectory, and (c) a reference position that is moving circularly through the simulation. Additionally, is also presented two sets of results regarding the limitations of the algorithm: (a) the input of an initial state that does not respect the physical limitations of the Niryo One robotic arm and (b) the input of an unreachable reference position.

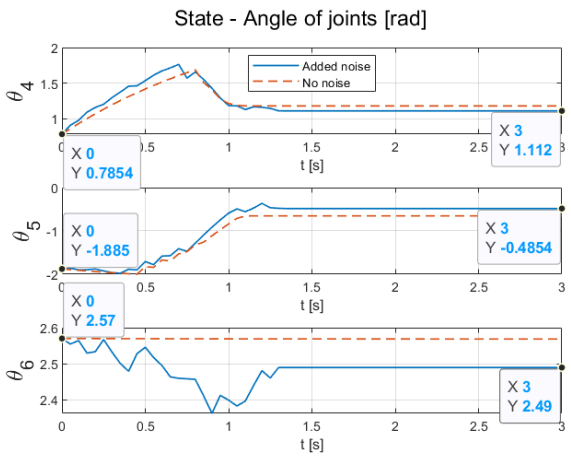
Each set of results consists of the progression of the state (the angle of each joint), the control (the desired angles for the joints), the velocity over time, and the evolution of the position of the end-effector during the simulation. In all the graphics except the control signal one, with a solid blue line it is represented the simulation with added noise, and with a black dotted line or a cross-marker it is represented the reference position of the end-effecor. It is also presented the simulation without the added noise during the trajectory with an orange dashed line.

5.1 Fixed reference position

In this scenario, it was given a valid initial state of $x = (-3, 0.5, 0, \frac{\pi}{4}, -\frac{3\pi}{5}, 2.57)$, and a reachable reference position of $y^e_{ref} = (95.6238, -245.0072, 218.3647)$ that corresponds to an angle of $\frac{\pi}{8}$ radians for all joints.

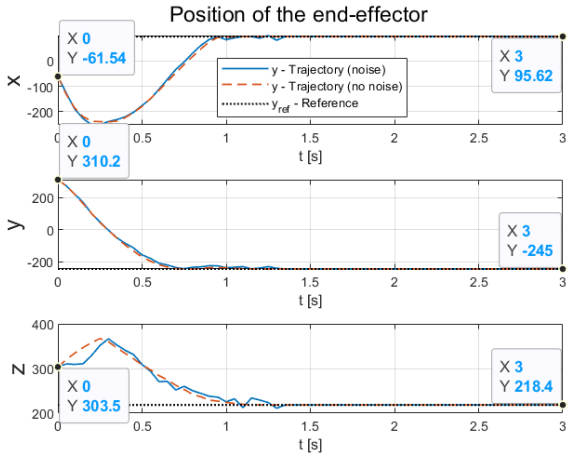


(a) J1 to J3 - scenario 1

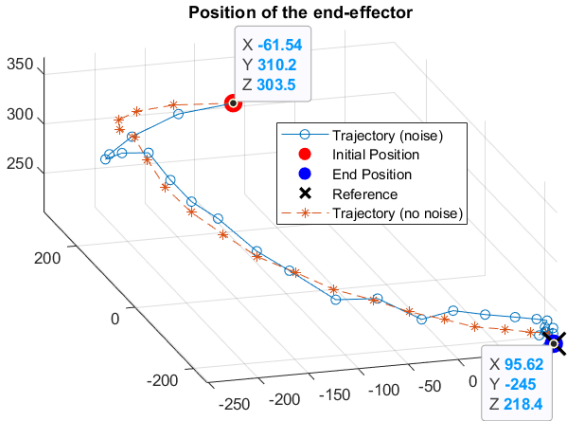


(b) J4 to J6 - scenario 1

Figure 5.1: Angle of each joint - scenario 1



(a) Coordinates' evolution - scenario 1



(b) Position of the end-effector (J6) in the world frame - scenario 1

Figure 5.2: Position of the end-effector (J6) - scenario 1

It can exist several combinations of joints' angles that lead to the identical position of the end-effector, and it is possible to observe in Figure 5.1 and in Figure 5.2 that the MPC algorithm reaches the same position with other angles - $x = (0.351, 0.3659, 0.4748, 1.112, -0.4854, 2.49)$ - at the end, set to 3 seconds, of both simulations, with and without noise. The problem of computing the set of joint angles in order to achieve a desired position and orientation of the end-effector is the inverse kinematics [2], which is not approached in this dissertation, noting that the MPC algorithm is able to provide a solution in each simulation result.

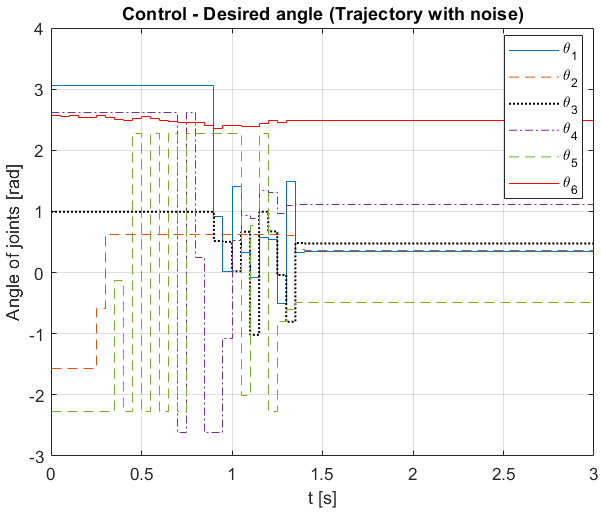
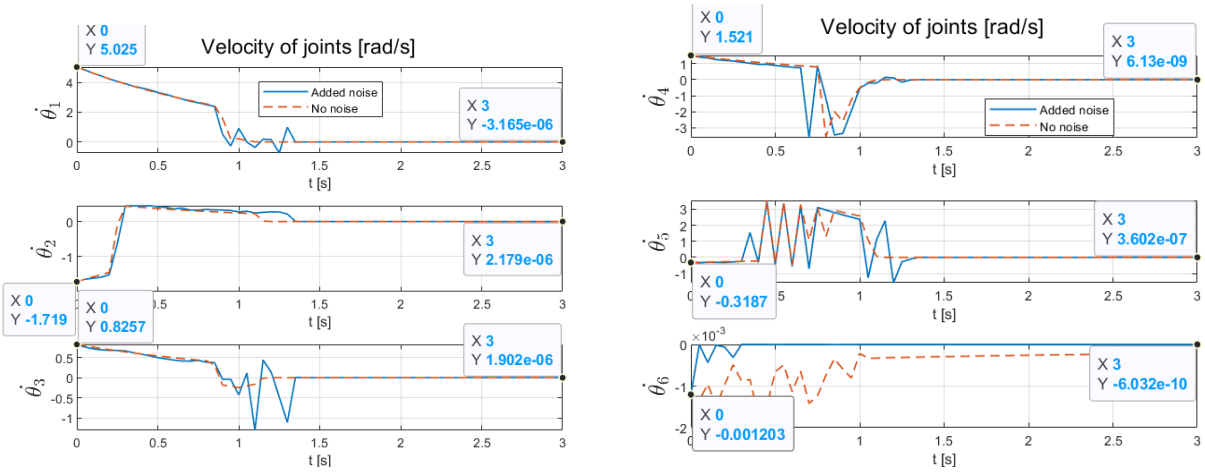


Figure 5.3: Control (desired angle) for each joint - scenario 1

The model of the robotic arm reaches the reference position, and its joints converge after around 1.5 seconds, as can be seen in Figure 5.2(a), and when the control signal stops varying in Figure 5.3. The variation of the angles between two consecutive iteration, the velocity and consequently the time that it takes to reach the reference position depend on the model parameters discussed in Chapter 3.



(a) Velocity J1 to J3 - scenario 1

(b) Velocity J4 to J6 - scenario 1

Figure 5.4: Velocity of the joints - scenario 1

In Figure 5.4, it is possible to notice that the velocity of each joint is increasing or decreasing at a constant rate throughout most of the trajectory, only changing in slope or direction when the control signal changes, as can be seen in Figure 5.3. Focusing on the latest, the desired angles for each joint are step signals that change while the MPC adapts and tries to find the best path to reach the desired reference position. Analyzing the state of the joints, the velocity, and the position of the end-effector, it can be confirmed that the robotic arm moves smoothly throughout most of the trajectory and reaches the expected reference position, even when noise is added to the simulation.

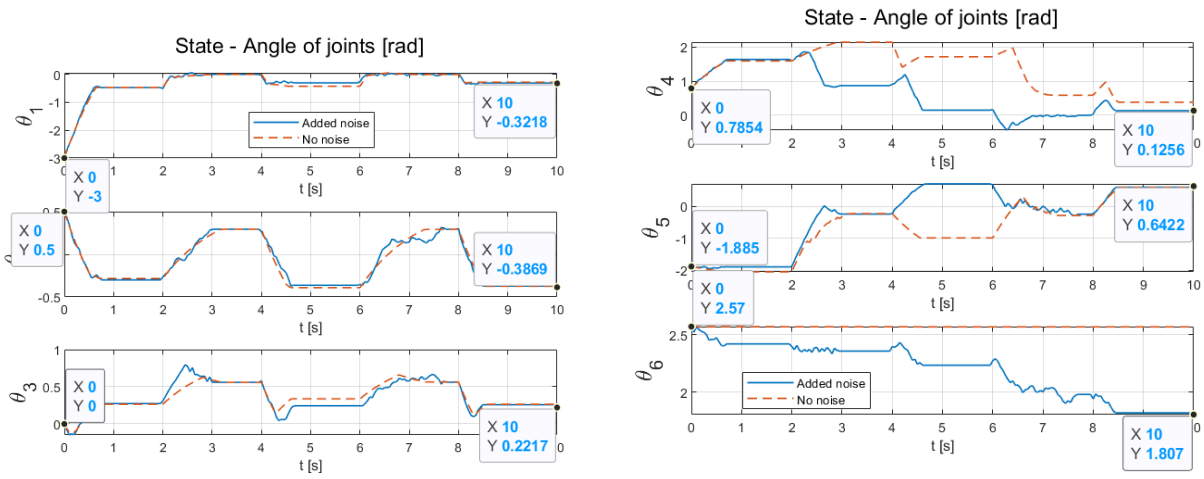
Another scenario of a successful result with a fixed reference position is provided in Appendix A.

5.2 Two variable reference positions

One of the main benefits of using the MPC method to control the robotic arm is the aptitude to change its parameters during the working process. In the particular case of this project, the possibility of changing the reference position during the trajectory of the end-effector, and not having a fixed one is very important to meet the objective of applying this control technique to a robotic arm used for assistive feeding purposes since the mouth of the person being fed (the reference position) is movable in the world space.

Two different cases where the reference position is variable are provided. The results show the success in following and reaching the reference while maintaining a relatively smooth movement of the joints through most of the trajectory.

In this second simulation scenario, two different reference positions alternate every two seconds of the simulation. For a better understanding of the results, the simulation time was increased to ten seconds.



(a) J1 to J3 - scenario 2

(b) J4 to J6 - scenario 2

Figure 5.5: Angle of each joint - scenario 2

In Figure 5.5, one can notice that the addition of noise in the position of the end-effector does not affect much the joint's movement for the first three joints. It is also possible to see that the angle of each joint varies, as expected, in a cyclic manner with slight changes between each period.

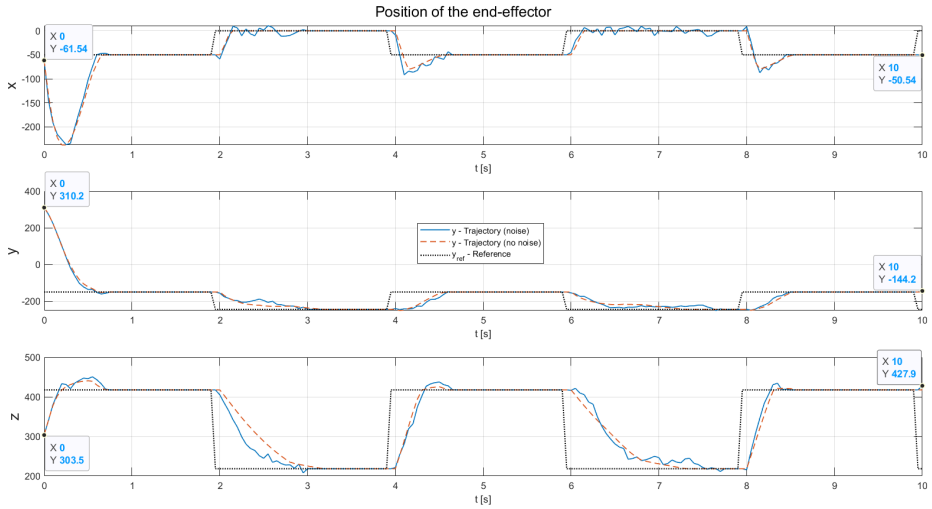


Figure 5.6: Coordinates' evolution - scenario 2

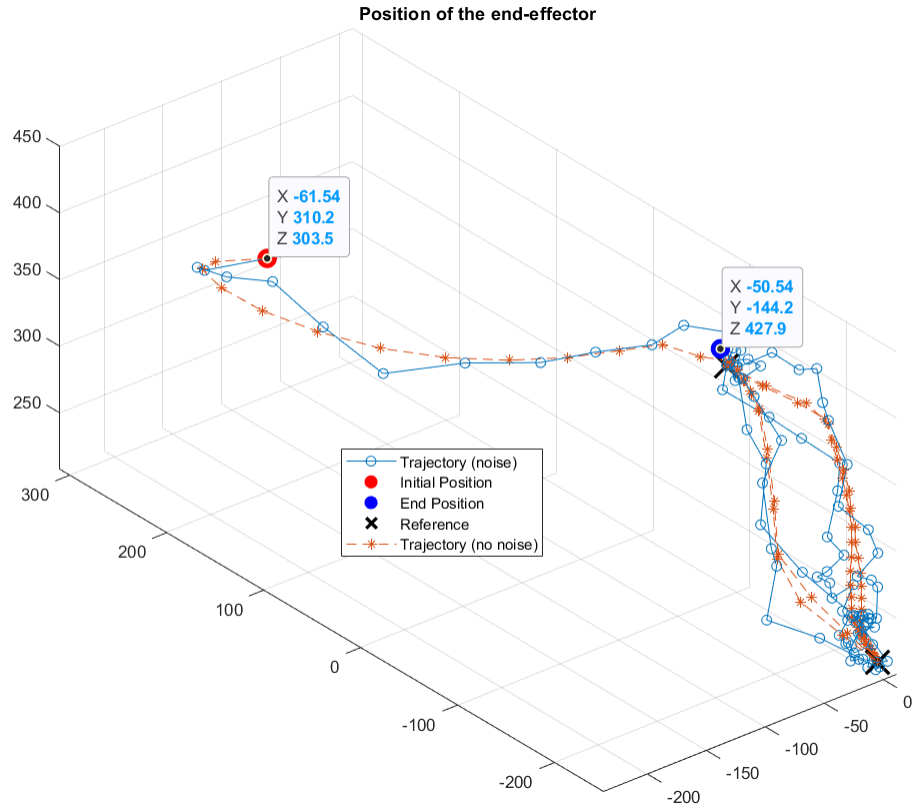
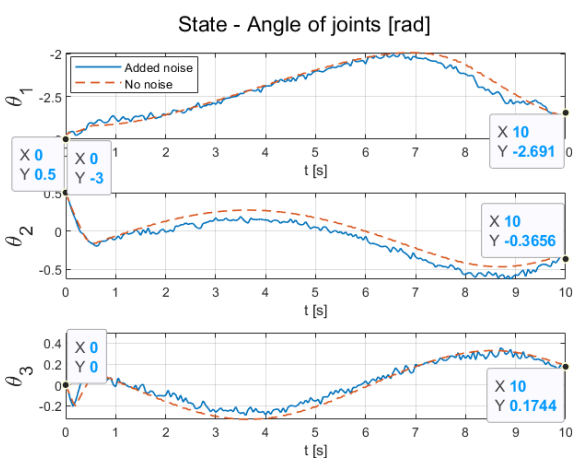


Figure 5.7: Position of the end-effector (J6) in the world-frame - scenario 2

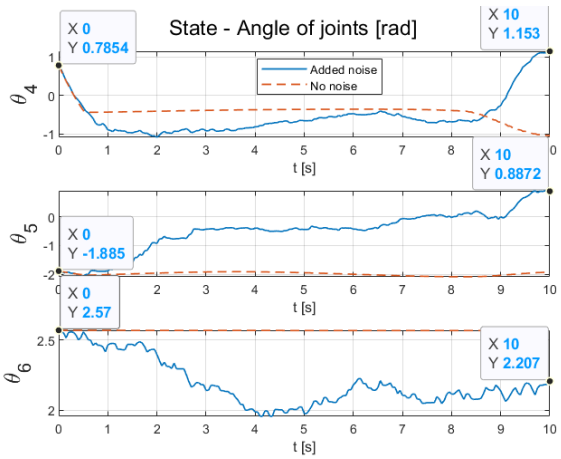
As can be observed in Figure 5.6 and Figure 5.7, the reference position is followed by the model of the robotic arm, and the algorithm adapts quite quickly to the change in the reference position without making abrupt variations in the movement of the end-effector. Although the angles of J4 to J6 are different during the simulation, as presented in Figure 5.5, the trajectory of the end-effector when the noise is added is very similar to when it is not.

5.3 Circular reference positions

The results for the third scenario in which the reference position changes through time are presented in this section, where the reference trajectory of the end-effector describes a circumference.



(a) J1 to J3 - scenario 3



(b) J4 to J6 - scenario 3

Figure 5.8: Angle of each joint - scenario 3

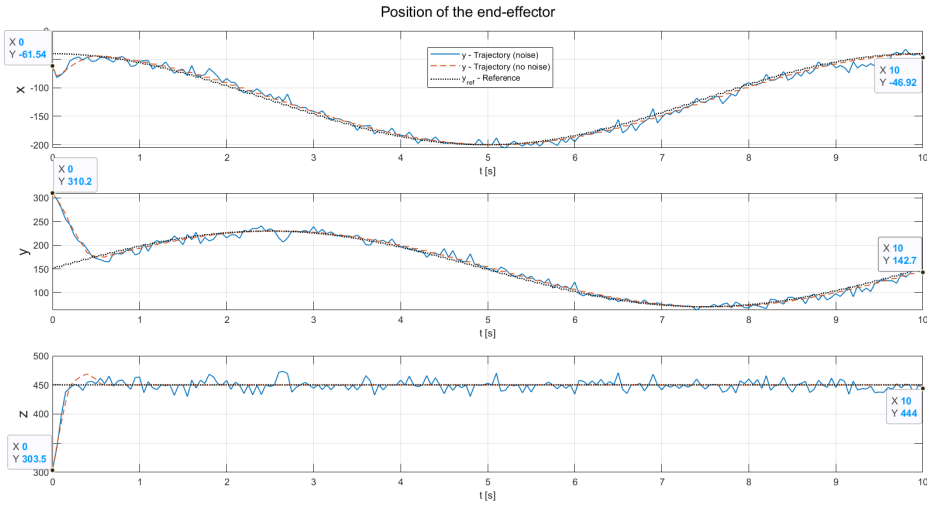


Figure 5.9: Coordinates' evolution - scenario 3

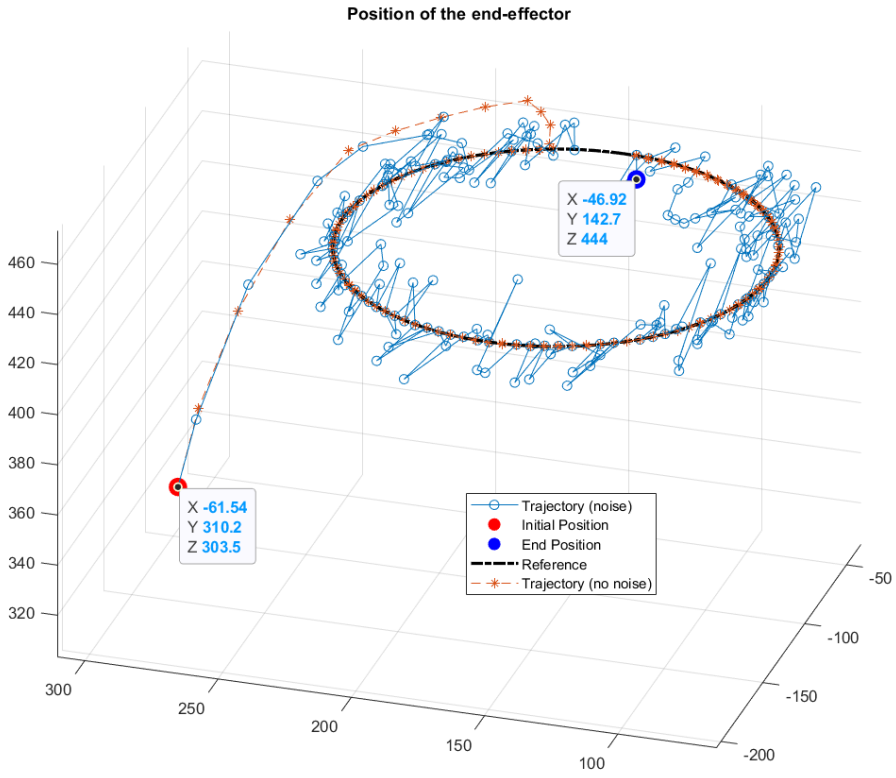


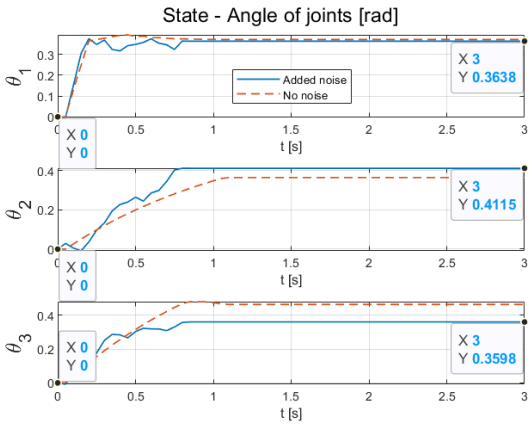
Figure 5.10: Position of the end-effector (J6) in the world-frame - scenario 3

In Figure 5.8, Figure 5.9 and Figure 5.10, one can notice that the end-effector follows the circular reference, although the precise position is not met because the MPC does not have time to settle once the reference changes in every iteration given the addition of noise in the current position of the arm. For the simulation without the added noise, once the end-effector reaches a desired reference it follows its trajectory flawlessly.

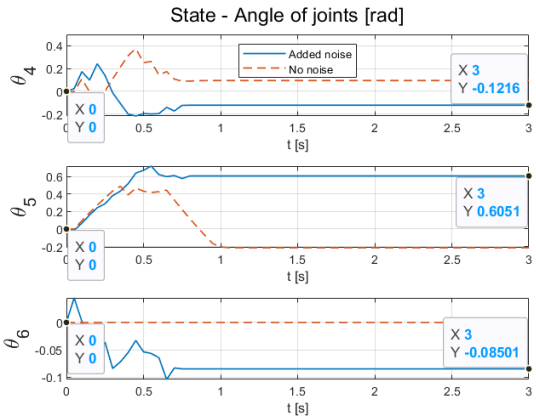
5.4 Invalid initial state

In this scenario, it was given an input that does not respect the physical limitations of the robotic arm and, consequently, the constraints defined in the optimization problem. The initial state provided is $x_0 = (-5, -5, 0, \frac{\pi}{4}, 0, 0)$ where the angles for J1 and J2 are invalid.

It can be observed in Figure 5.11 that the MPC algorithm automatically sets the initial state to a valid one, $x_0 = (0, 0, 0, 0, 0, 0)$, and proceeds to define the trajectory for the end-effector normally.



(a) J1 to J3 - scenario 4



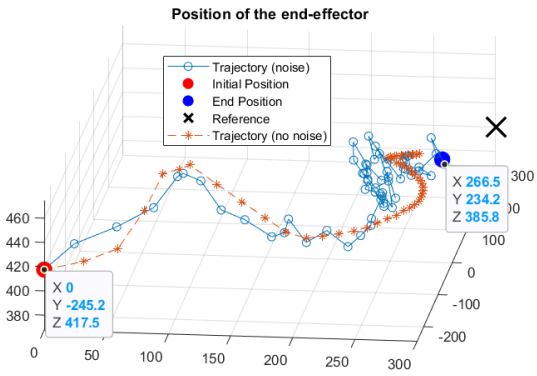
(b) J4 to J6 - scenario 4

Figure 5.11: Angle of each joint - scenario 4

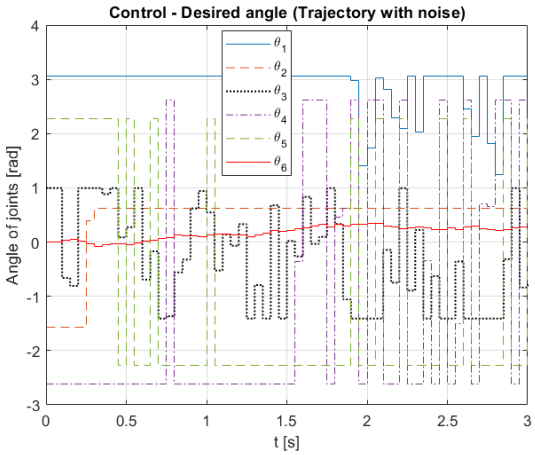
5.5 Invalid reference position

The results for the last scenario, where an invalid fixed reference position is given, are presented in this section. It was given a reference position of $y^e_{ref} = (500, 500, 500)$, that can not be reached with any combination of angles for the joints respecting the physical limitations of the robotic arm for the provided forward kinematics' scheme.

It can be observed in Figure 5.12 that the end-effector does not reach that location, but tends to the last reachable position in the trajectory.



(a) Position of the end-effector (J6) - scenario 5



(b) Control (desired angle) for the joint - scenario 5

Figure 5.12: Invalid reference position - scenario 5

6

Conclusions and future work

In this dissertation, the structure of a six-DOF robotic manipulator was studied to the extent of reaching the main objective that was to design an algorithm based on the MPC method that solves Optimal Control Problems (OCP) to plan the trajectory for the end-effector to a fixed or variable reference, adapting in real-time to disturbances in its position.

The geometrical properties of the manipulator were analyzed to compute the forward kinematics, the correspondence between the angles of the six joints of the arm, and the position of the end-effector in a base reference frame. Since it was not possible to establish a complete dynamic model for the robotic arm, as explained in Section 3.3, a simpler differential system's model was defined to represent it in the simulation, and its parameters were estimated from the response to simple commands.

Based on the simulation results, it is confirmed that MPC satisfies the goal of reaching the reference position, whether it is fixed or variable through time, reacting to the perturbations imposed by the addition of noise to the end-effector in each iteration of the algorithm while respecting the physical constraints of the manipulator and while generating and maintaining a relatively steady trajectory. It is possible to observe that when no noise is added to the position of the end-effector, the trajectory is cleaner, but its presence does not significantly affect the smoothness of the joints' movement nor the performance of the algorithm.

As expected, the MPC method proved to be a powerful tool for robotic manipulation given the capacity to handle constraints, multiple inputs and outputs, and the possibility for the user to define the cost function. Although only the simulation results are provided, these features and the results presented in this thesis represent an achievement for assistive and feeding purposes since users' mouth is not fixed in the manipulator's workspace and can still be reached using this method.

In the future, the work made and presented in this document can be expanded. The first step would be to adapt, for example, the algorithm's parameters and the iterations' frequency in order to execute it in reality and in real-time, in the Niryo One robotic arm, to have a full spectrum of results both in simulation and in experimental work. With these experiments, it would be possible to verify if the simplified model used in the simulation is enough to satisfy the requirements of the feeding task or if a full model of the arm is necessary to improve the results. Even if not required, the complete dynamic model of a six-DOF manipulator would be an interesting topic of study. Also, the parameters' identification using formal techniques might be required when applying the algorithm to the concrete ARM to achieve satisfactory trajectories.

Once the MPC is functioning successfully in reality, vision and machine learning techniques, such as deep and reinforcement learning, can be used to improve the algorithm by integrating with the optimization the visual information of the environment, like the individuals' expressions while eating.

Bibliography

[1] W. Khalil and E. Dombre, “Chapter 1 - terminology and general definitions,” in *Modeling, Identification and Control of Robots*, W. Khalil and E. Dombre, Eds. Oxford: Butterworth-Heinemann, 2002, pp. 1–12. [Online]. Available: https://doi.org/10.1016/B978-190399666-9/50001-4

[2] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Pearson Education International, 2005. [Online]. Available: http://www.mech.sharif.ir/c/document_library/get_file?uuid=5a4bb247-1430-4e46-942c-d692dead831f&groupId=14040

[3] Automation strategies drive 12% increase in number of robots at work globally. Accessed 10-October-2021. [Online]. Available: https://ifr.org/post/automation-strategies-drive-12-increase-in-number-of-robots-at-work-globally

[4] A. Blom and H. Stuyt, *3. Assistive Robotic Manipulators: Principles and Practice*. CRC Press, 03 2017, pp. 71–98. [Online]. Available: https://doi.org/10.4324/9781315368788-3

[5] R. K. Al-Halimi and M. Moussa, “Performing complex tasks by users with upper-extremity disabilities using a 6-dof robotic arm: A study,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 6, pp. 686–693, 2017. [Online]. Available: https://doi.org/10.1109/TNSRE.2016.2603472

[6] M. Topping, “An overview of the development of handy 1, a rehabilitation robot to assist the severely disabled,” *Journal of Intelligent and Robotic Systems*, vol. 34, pp. 253–263, 07 2002. [Online]. Available: https://doi.org/10.1023/A:1016355418817

[7] A. Candeias, T. Rhodes, M. Marques, J. P. Costeira, and M. Veloso, “Vision augmented robot feeding,” in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, pp. 50–65. [Online]. Available: https://doi.org/10.1007/978-3-030-11024-6_4

[8] V. Balaji and D. L. Rajaji, “Comparative study of pid and mpc controller using lab view,” *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2013. [Online]. Available: https://www.ijareeie.com/upload/2013/november/17_COMPARATIVE.pdf

[9] Niryo One - Description and mechanical specifications. Accessed 10-July-2021. [Online]. Available: https://niryo.com/product/niryo-one/

[10] D. Feil-Seifer and M. Mataric, “Defining socially assistive robotics,” in *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.*, 2005, pp. 465–468. [Online]. Available: https://doi.org/10.1109/ICORR.2005.1501143

[11] A. S. Prabuwono, K. Allehaibi, and K. Kurnianingsih, “Assistive robotic technology: A review,” *Computer Engineering and Applications Journal*, vol. 6, pp. 71–78, 07 2017. [Online]. Available: https://www.researchgate.net/publication/328405471_Assistive_Robotic_Technology_A_Review

[12] U. Nations, *World population ageing, 2019 : highlights*. New York :. 2019: UN,, 2019, includes bibliographical references (p. 29-30). [Online]. Available: https://doi.org/10.18356/9df3caed-en

[13] S. Ishii, S. Tanaka, and F. Hiramatsu, “Meal assistance robot for severely handicapped people,” in *Proceedings of 1995 IEEE International Conference on Robotics and Automation*, vol. 2, 1995, pp. 1308–1313 vol.2. [Online]. Available: https://doi.org/10.1109/ROBOT.1995.525461

[14] P. Lopes, R. Lavoie, R. Faldu, N. Aquino, J. Barron, M. Kante, B. Magfory, and W. Meleis, “Eye-controlled robotic feeding arm technology (icraft),” 2012, p. 1–24. [Online]. Available: https://ece.northeastern.edu/personal/meleis/icraft.pdf

[15] G. Chance, A. Camilleri, B. Winstone, P. Caleb-Solly, and S. Dogramadzi, “An assistive robot to support dressing - strategies for planning and error handling,” *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 774–780, 06 2016. [Online]. Available: https://doi.org/10.1109/BIOROB.2016.7523721

[16] S. Schröer, I. Killmann, B. Frank, M. Völker, L. Fiederer, T. Ball, and W. Burgard, “An autonomous robotic assistant for drinking,” *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015, pp. 6482–6487, 06 2015. [Online]. Available: https://doi.org/10.1109/ICRA.2015.7140110

[17] J. Vongkulbhisal, F. D. la Torre, and J. Costeira, “Discriminative optimization: Theory and applications to computer vision,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 41, no. 04, pp. 829–843, apr 2019. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TPAMI.2018.2826536

[18] J. Rawlings and D. Mayne, *Model Predictive Control: Theory, Computation and Design*, 2nd ed. Nob Hill Publishing, LLC, 01 2009. [Online]. Available: https://sites.engineering.ucsb.edu/~jbraw/mpc/

[19] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of nonlinear model predictive path-following control for an industrial robot," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1505–1511, 2017. [Online]. Available: https://doi.org/10.1109/TCST.2016.2601624

[20] N. van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, "Path-following nmpc for serial-link robot manipulators using a path-parametric system reformulation," in *2016 European Control Conference (ECC)*, 2016, pp. 477–482. [Online]. Available: https://doi.org/10.1109/ECC.2016.7810330

[21] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and fast tracking control on a robot manipulator: Robust MPC and neural network control," *CoRR*, vol. abs/1912.10360, 2019. [Online]. Available: https://doi.org10.1109/LRA.2020.2975727

[22] P. Drews, G. Williams, B. Goldfain, E. Theodorou, and J. Rehg, "Aggressive deep driving: Model predictive control with a cnn cost model," 07 2017. [Online]. Available: https://www.researchgate.net/publication/318489175_Aggressive_Deep_Driving_Model_Predictive_Control_with_a_CNN_Cost_Model

[23] C. Alavala, "Difference between denavit-hartenberg (d-h) classical and modified conventions for forward kinematics of robots with case study," December 2014. [Online]. Available: https://doi.org/10.13140/2.1.2012.9607

[24] W. Khalil, "Dynamic modeling of robots using newton-euler formulation," *Lecture Notes in Electrical Engineering*, vol. 89, pp. 3–20, 01 2011. [Online]. Available: https://doi.org/10.1007/978-3-642-19539-6_1

[25] S. Dong, Z. Yuan, and F. Zhang, "A simplified method for dynamic equation of robot in generalized coordinate system," *Journal of Physics: Conference Series*, vol. 1345, p. 042077, 11 2019. [Online]. Available: https://doi.org/10.1088/1742-6596/1345/4/042077

[26] L. Ding, H. Wu, Y. Yao, and Y. Yang, "Dynamic model identification for 6-dof industrial robots," *Journal of Robotics*, vol. 2015, pp. 1–9, 10 2015. [Online]. Available: https://doi.org/10.1155/2015/471478

[27] Niryo One - MATLAB interface. Accessed 10-July-2021. [Online]. Available: https://github.com/NiryoRobotics/niryo_one_matlab

[28] C. E. García, D. M. Prett, and M. Morari, “Model predictive control: Theory and practice—a survey,” *Automatica*, vol. 25, no. 3, pp. 335–348, 1989. [Online]. Available: https://doi.org/10.1016/0005-1098(89)90002-2

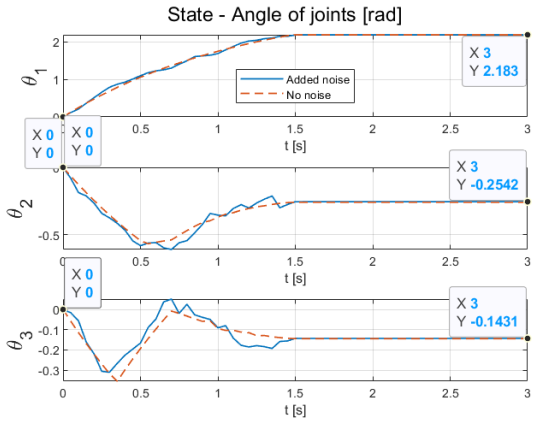
[29] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, “acados: a modular open-source framework for fast embedded optimal control,” *Mathematical Programming Computation*, 2021. [Online]. Available: https://doi.org/10.1007/s12532-021-00208-8

[30] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi – A software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019. [Online]. Available: https://doi.org/10.1007/s12532-018-0139-4

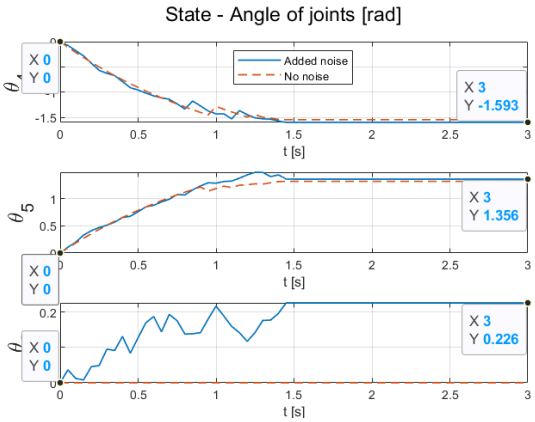
[31] acados - Problem Formulation OCP. Accessed 10-July-2021. [Online]. Available: https://github.com/acados/acados/blob/master/docs/problem_formulation/problem_formulation_ocp_mex.pdf

A

Set of results: Fixed reference position



(a) J1 to J3 - scenario 6



(b) J4 to J6 - scenario 6

Figure A.1: Angle of each joint - scenario 6

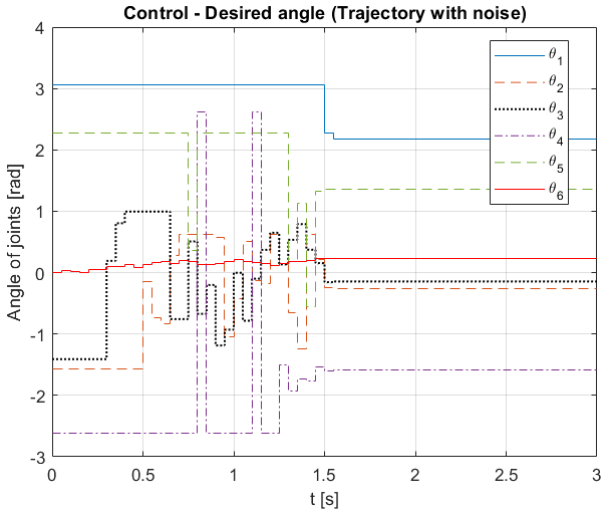
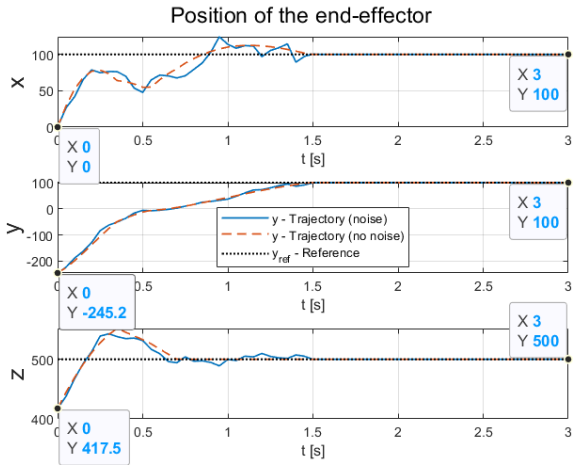
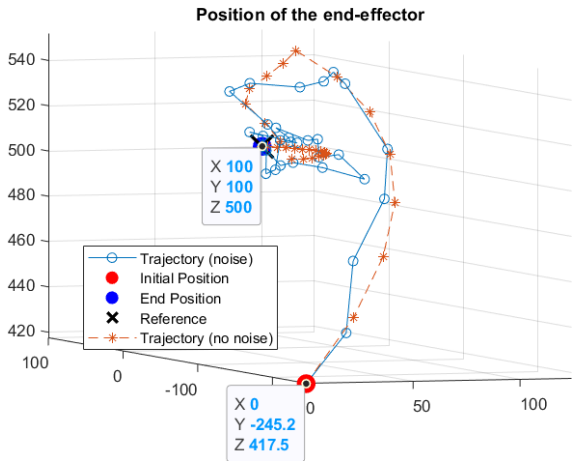


Figure A.2: Control (desired angle) for each joint - scenario 6

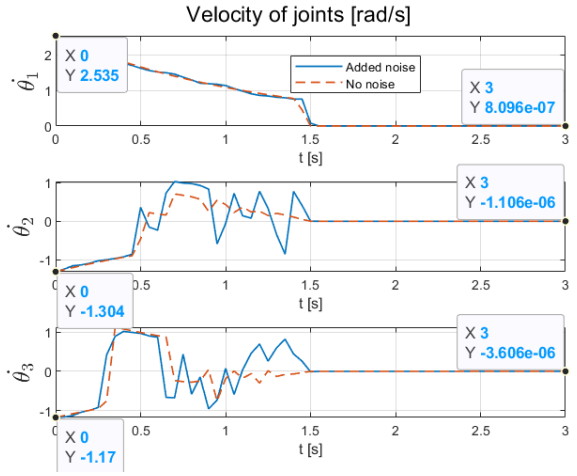


(a) Coordinates' evolution - scenario 6

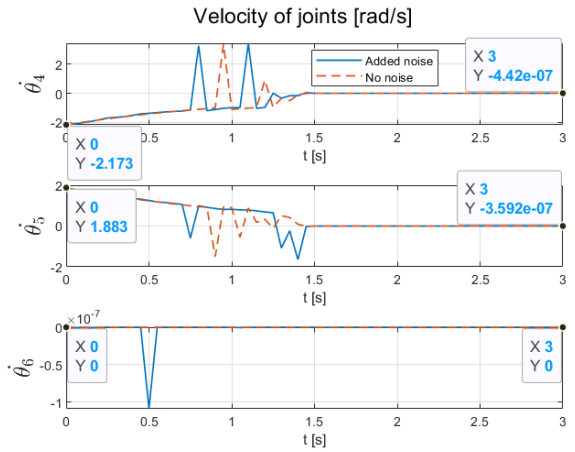


(b) Position of the end-effector (J6) in the world frame - scenario 6

Figure A.3: Position of the end-effector (J6) - scenario 6



(a) Velocity J1 to J3 - scenario 6



(b) Velocity J4 to J6 - scenario 6

Figure A.4: Velocity of the joints - scenario 6