الجمـــهوريـــة الجـــزائريـــة الديـمقراطيـــــة الشـعبيـــــة
THE PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
وزارة التعـلـيـم الـعـالـي والـبـحـث الـعـلـمي
THE MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
جـامـعـة عمّـار ثـلـيـجـي بالأغـــواط
AMAR TELIDJI UNIVERSITY OF LAGHOUAT

كـلـيـــة الـتكنولوجيا
FACULTY OF TECHNOLOGY

قسم الالكتروتقني
DEPARTMENT OF ELECTROTECHNIC

Master's thesis

Domain:	Sciences and Technologies
Field:	Automatic
Option:	Automatic and systems

Theme

Modelling and Control of Mobile Manipulator

Presented by:
Mr. Ali Choucha
Mr. Fathi Madjour

Publicly supported ahead of a jury composed of:

Mr. BENMOUIZA Khalil	*M.C.A*	*President*
Mr. OUBBATI Khalil Ibrahim	*M.C.B*	*Examiner*
Mr. RAHMANI Belkacem	*M.C.B*	*Supervisor*

Academic year 2021/2022

Abstract

The rising number of robot manipulators required to execute a job has necessitated the development of complex control algorithms. However, because the efficiency and stability of these schemes are the primary concerns, as a result, the focus has shifted away from developing accurate and precise kinematics and dynamics models. Such Models are the foundation of every control method since they give fundamental information. About the system's characteristics the youBot mobile manipulator from KUKA is a manipulator installed on a transportable platform designed for educational and research applications Because of its kinematics, the five (low)-DOF manipulator is an intriguing case study and in fact it’s difficult. Furthermore, the omnidirectional platform on which the manipulator is placed is an intriguing instance itself. The purpose of this thesis is to create a kinematic model for the KUKA youBot in two parts: a separated system (mobile platform model and a manipulator model) and combined system, then controlling the obtained models using MATLAB.

Key words: Kuka YouBot - The Omnidirectional Platform - Manipulator- 5DOF Manipulator - Mobile robot - Kinematics and Dynamics Models.

ملخص

استلزم العدد المتزايد من الروبوتات المناورة المطلوبة لتنفيذ وظيفة، تطوير خوارزميات تحكم معقدة. ومع ذلك، نظرًا لأن كفاءة واستقرار هذه المخططات هي الاهتمامات الأساسية، ونتيجة لذلك، فقد تحول التركيز الى تطوير نماذج حركية وديناميكية دقيقة. هذه النماذج هي أساس كل طريقة تحكم لأنها تقدم معلومات أساسية حول خصائص النظام. يعتبر الروبوت المناور المتحرك "YouBot" من شركة KUKA ذراعا مناورا مثبتًا على قاعدة أساسية قابلة للنقل، حيث صمم هذا الروبوت للتطبيقات التعليمية والبحثية نظرًا لخصائصه الحركية، يعد الذراع المناور الذي يأتي بخمسة درجات من الحرية حالة دراسة مثيرة للاهتمام وصعبة. علاوة على ذلك، فإن القاعدة الاساسية متعددة الاتجاهات التي يوضع عليها الذراع المناور هي مثال مثير للاهتمام بحد ذاته. الغرض من هذه الرسالة هو إنشاء نموذج حركي لـ KUKA youBot في جزأين: نظام منفصل (نموذج المنصة المتنقلة ونموذج الذراع المناور) ونظام مدمج، ثم التحكم في النماذج التي تم الحصول عليها باستخدام.MATLAB .

الكلمات المفتاحية: Kuka YouBot- المنصة متعددة الاتجاهات - مناور - 5 درجات من الحرية (DOF) - روبوت متحرك - نماذج حركية وديناميكية.

Résumé

Le nombre croissant de robots manipulateurs nécessaires à l'exécution d'un travail a nécessité le développement d'algorithmes de contrôle complexes. Cependant, comme l'efficacité et la stabilité de ces schémas sont les principales préoccupations, l'accent s'est détourné du développement de modèles cinématiques et dynamiques précis. Ces modèles sont à la base de toute méthode de contrôle car ils fournissent des informations fondamentales. À propos des caractéristiques du système, le manipulateur mobile youBot de KUKA est un manipulateur installé sur une plate-forme transportable conçue pour les applications éducatives et de recherche En raison de sa cinématique, le manipulateur à cinq (faibles) DDL est une étude de cas intrigante et en fait difficile. De plus, la plate-forme omnidirectionnelle sur laquelle le manipulateur est placé est un exemple intrigant en soi. Le but de cette thèse est de créer un modèle cinématique pour le KUKA youBot en deux parties : un système séparé (modèle de plate-forme mobile et modèle de manipulateur) et un système combiné, puis de contrôler les modèles obtenus à l'aide de MATLAB.

Mots clés : Kuka YouBot - La plateforme omnidirectionnelle – Manipulateur - 5-DDL Manipulateur - Robot mobile - Modèles cinématiques et dynamiques.

ACKNOWLEDGEMENTS

This work was made possible thanks to the help of several people to whom we would like to express our gratitude.

First of all, we would like to express our gratitude to the director of this work, Dr. Rahmani Belkacem, for his patience, his availability and above all his judicious advice, which helped to fuel our reflections.

We would also like to thank the professors at the Department of Electrotechnic, who provided us with the necessary tools for the success of this study.

We would like to express our gratitude to the friends and colleagues who have provided us with moral and intellectual support throughout our process.

Finally, we would like to express our gratitude to everyone supported us in this work.

DEDICATION

I dedicate my dissertation work to my family and many friends. A special feeling of gratitude to my loving parents, Kouider and F. Choucha whose words of encouragement and push for tenacity ring in my ears. My brothers and sisters, Messoud, Mohamed, Toufik, Abd Elrazak, Hadjer and janaa. They have never left my side and are very special.

I also dedicate this work to all the aunts, uncles and grandparents, may God prolong their life.

I also dedicate this dissertation to my many friends and church family who have

supported me throughout the process. I will always appreciate all they have done, especially Boumadian baguira, Zin El Abidin Mihoubi, Mohamed Boubakraui and Fathi.

Choucha Ali

DEDICATION

I dedicate my dissertation work to my family and many friends. A special feeling of gratitude to my loving parents, Rahmani Aek and Denia whose words of encouragement and push for tenacity ring in my ears. My brother ADEM who have never left my side.

I also dedicate this work to all the aunts, uncles and grandparents, may God prolong their life.

I also dedicate this dissertation to my many friends and church family who have

supported me throughout the process. I will always appreciate all they have done, especially Hamza, Mohsen.
Mahmoud and Ali.

Madjour Fathi

Contents

List of Figures

List of Abbreviations

MATLAB: Math Laboratory.

ROS: Robot Operating System.

DOF: Degrees of freedom.

List of Tables

List of Symbols

T_{sb}: representing a chassis-fixed frame $\{b\}$ relative to a fixed space frame $\{s\}$ in the horizontal plane.

q: three coordinates (ϕ, x, y) of the chassis.

$\dot{q}$: the velocity of the chassis $(\dot{\phi}, \dot{x}, \dot{y})$.

$\hat{\mathrm{x}}_{\mathrm{w}} - \hat{\mathrm{y}}_{\mathrm{w}}$: the frame at the center of the wheel.

v: the linear velocity of the center of the wheel.

v_{drive} : is the driving speed.

v_{slide} : is the sliding speed.

r: the radius of the wheel.

u: the driving angular speed of the wheel.

v_b: the body twist.

ϕ: the chassis orientation.

(β_i, x_i, y_i): The wheel positions and headings.

γ_i: denotes the angle at which free "sliding" occurs on wheel i.

ℓ: is the distance between the chassis center and wheels axis.

w: is the distance between the wheel center and wheels axis center.

a_i: the distance between the origins of the two coordinate frames O_i, $O_{i'}$.

d_i: the coordinate of O_i along z_{i-1}.

α_i: the angle between axes z_{i-1} and z_i about axis x_i to be taken positive when rotation is made counter-clockwise.

θ_i: the angle between axes x_{i-1} and x_i about axis z_{i-1} to be taken positive when rotation is made counter-clockwise.

a: is the distance between the arm's two vertical z-axes.

${}^{i-1}\mathrm{T}_i$: The homogeneous transformation from frame i to frame $i-1$.

$\mathcal{T}$: Kinetic energy of the system.

$\mathcal{U}$: Potential energy of the system.

q_i: $-^{\text{th}}$ generalized system coordinate.

τ_i: Generalized force applied to the 1st element.

$M(q)$: is the inertia matrix.

$C(q, \dot{q})$: is the Coriolis forces matrix.

$G(q)$: is the gravitational forces vector.

J^*: the composite Jacobian of the combined system.

J_L^*: the end-effector's linear speeds.

J_A^*: the end-effector's rotational velocities.

K_p: proportional action gain.

K_i: integral action gain.

K_d: derivative action gain.

General Introduction

General Introduction

Several years ago, there were many inventions and developments in the field of technology. All these developments are manufactured things that are of great benefit to mankind. Among them are the developments that have caused controversy and a boom in the world in the technological field is robots. According to the known information about them, they are programmed machines to do the planned work, but in order to understand and deepen the way in which they work, we studied one of these robots from designing mathematical equations that represent them to simulating them by using a robot simulation program.

The first chapter touches an introduction that will help us to take an image about the development of industry robots. It covers also the chosen Kuka youBot robot components, characteristics and uses. Moreover, we studied the modeling of Kuka youBot in two main parts. In the first part our Kuka youBot robot modeling was done in two sections: the first step was the mobile platform modeling and the second step was the manipulator modeling. In the platform modeling section, a general definition and some characteristics was given before starting developing the kinematic model step by step. Also, in the manipulator modeling section, we start by developing the kinematics down to giving the dynamic modeling laws. In the second part, by assuming that the arm (manipulator) is mounted on the platform, the kinematic model (forward and differential kinematics) for our Kuka youbot robot was developed and studied in detail.

In the Second chapter, the control part was done using MATLAB software, definition of trajectory technic was given, down to the simulation section that was done in two parts: controlling the separated system then controlling the combined system in order to check the correctness of the models in each of them. In the other hand, the manipulator control in the separated system was done using three controllers to choose the best one.

At the end of these chapters, a general conclusion was given as well as future work.

Chapter 1 Modeling of Kuka YouBot

1.1 Introduction

With the development of technology and human needs from every aspect, the increase in factories has become great to provide what the individual needs in his daily life, and thus the demand for labor increases with the diversity of the risks of these factories, so the idea of inventing robots is to perform human tasks and reduce the workforce to avoid risks. For several decades, robots have been used in industry, mostly of the manipulator type, they are used to perform repetitive tasks in the production cycle. Another famous type is the Self-guided vehicles (also called mobile robots) are used in the manufacturing industry. However, research in the field of robotics is still very active and a very large and it is directed towards the development of new applications for robotics. The question is, how these robots are designed and simulated (using the mathematical modeling) before they are built?

There are a lot of industry robot types we have chosen the Kuka youBot mobile manipulator to study it and take as example. In this chapter, we will talk about the components, characteristics and uses of our Kuka youBot robot, along with the mathematical modeling.

1.2 Kuka youBot

The Kuka youbot is a mobile manipulator designed primarily for educational and scientific purposes. Many laboratories and institutes throughout the world that specialize on the development and testing of new robotic technology are gradually adopting it as a standard platform. Such a system allows users to access and experiment with fundamental robot functions at a reasonable cost. A brief description of the mobile manipulator Let us begin by looking at the Kuka youbot: [3]

• An omnidirectional mobile platform that consists of the robot chassis, four mecanum wheels, motors, power and an onboard PC board. Users have two options: This board can run applications or being controlled remotely from a computer. [3]

• A robotic arm with five degrees of freedom (DOF) and a two-finger gripper. [3]

A robotic platform like this has a lot of application potential. However, its architecture limits its capabilities. A manipulator that has fewer than six degrees of freedom (DOF) is incapable of positioning, and efficiently positioning an object for specialized industrial applications however, such as A low-DOF manipulator may be sufficient for welding, painting, and loading/unloading. Theoretically. What is a low-DOF manipulator's advantage over a high-DOF manipulator? [3]

The advantage of low-DOF robots over 6-DOF or redundant robots is that they have a simpler mechanical structure (i.e., less moving parts). motors and linkages), a more straightforward controller, increased rigidity, and a reduced cost as a result, the exponential growth in utilization is completely deserved.[3]

1.2.1 Kuka youBot characteristics

The Kuka youBot showing in Figure 1.1 has the following characteristics [4]:

- Omnidirectional mobile platform.
- Arm, 5-DOF manipulator: It is considered one of the main fixed components in any robot.
- Two-finger gripper: also, the main components fixed.
- Real-time Ether CAT communication.
- Open interfaces.
- freely programmable.

Figure 1.1: Kuka YouBot

Note: more characteristics and details of the Kuka youBot can be found in **Appendix (2).**

1.2.2 Kuka youbot utilities

We have many utilities of the Kuka youbot we mention [4]:

- Education.
- Basic research.
- Application development to use it in factories.

And the big model of Kuka youBot is used in different industry fields.

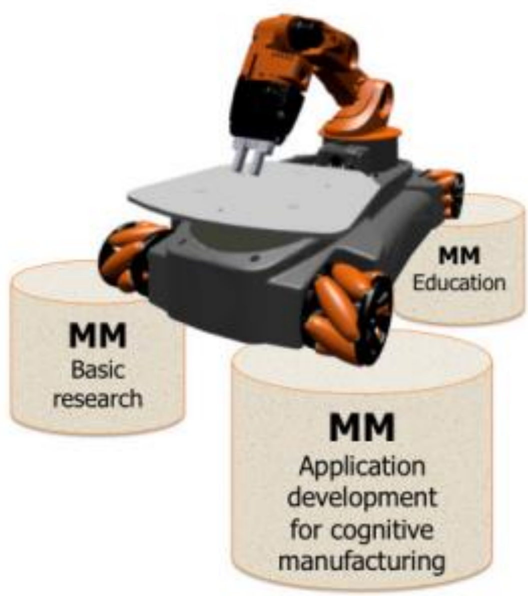

Figure 1.2: Kuka youbot utilities [4]

1.3 Modeling

Modeling can help us better visualize the plan of our system and allow us to develop more rapidly by helping us build the right thing [5]. In this chapter we will study the kinematic and dynamic modeling for our Kuka youbot robot in an independently form (modeling the mobile platform then modeling the manipulator), and in a combined form (assuming arm mounted on the platform).

1.3.1 Mobile Platform kinematics

1.3.1.1 Definition of a mobile robot

A mobile robot is a mechanical, electronic and computing system that physically acts with its environment to achieve an assigned purpose. The machine is versatile and adaptable to varying operating conditions. It has perception, decision and action functions. In this way, the robot should be able to perform various tasks, in several ways, and perform its task properly, even if it encounters unexpected new situations. [6]

At present, the most sophisticated mobile robots are mainly focused on applications in variable or uncertain environments, often with obstacles, requiring adaptation to the task. [6]

1.3.1.2 Omnidirectional mobile robots modeling

A kinematic model of a mobile robot governs how wheel speeds map to robot velocities, while a dynamic model governs how wheel torques map to robot accelerations. In this section, we ignore the dynamics and focus on the kinematics. We also assume that the robots roll on hard, flat, horizontal ground without skidding (i.e., tanks and skid-steered vehicles are excluded). The mobile robot is assumed to have a single rigid-body chassis (not articulated like a tractor-trailer) with a configuration $T_{sb} \in SE(2)$ representing a chassis-fixed frame $\{b\}$ relative to a fixed space frame $\{s\}$ in the horizontal plane. We represent T_{sb} by the three coordinates $q = (\phi, x, y)$. We also usually represent the velocity of the chassis as the time derivative of the coordinates, $\dot{q} = (\dot{\phi}, \dot{x}, \dot{y})$. [1] Occasionally it will be convenient to refer to the chassis' planar twist $\mathcal{V}_b = (\omega_{bz}, v_{bx}, v_{by})$expressed in {b}, where: [1]

$$\mathcal{V}_b = \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\phi & sin\,\phi \\ 0 & -sin\,\phi & cos\,\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} \tag{2.1}$$

$$\dot{q} = \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\phi & -sin\,\phi \\ 0 & sin\,\phi & cos\,\phi \end{bmatrix} \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix} \tag{2.2}$$

Figure 2.1: A mecanum wheel.

Our omnidirectional mobile robot has four mecanum wheels. That is what makes him achieve an arbitrary three-dimensional chassis velocity $\dot{q} = (\dot{\phi}, \dot{x}, \dot{y})$, since each wheel has only one motor (controlling its forward–backward velocity). Figure 2.2 shows the wheel motions obtained by driving the wheel motors as well as the free sliding motions allowed by the rollers. Two important questions in kinematic modeling are the following. [1]

(a) Given a desired chassis velocity $\dot{q}$, at what speeds must the wheels be driven? [1]

(b) Given limits on the individual wheel driving speeds, what are the limits on the chassis velocity $\dot{q}$? [1]

To answer these questions, we need to understand the wheel kinematics illustrated in Figure 2.3. In a frame $\hat{x}_w - \hat{y}_w$ at the center of the wheel, the linear velocity of the center of the wheel is written $v = (v_x, v_y)$, which satisfies [1]

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = v_{drive} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + v_{slide} \begin{bmatrix} -sin\,\gamma \\ cos\,\gamma \end{bmatrix} \tag{2.3}$$

Figure 2.2: the Kuka youBot mobile manipulator system, which uses four mecanum wheels for its mobile base. [1]

Where γ denotes the angle at which free “sliding” occurs (allowed by the passive rollers on the circumference of the wheel), v_{drive} is the driving speed, and v_{slide} is the sliding speed. for a mecanum wheel, typically γ = ±45". Solving Equation (2.3), we get [1]

$$\begin{aligned} v_{\text{drive}} &= v_x + v_y \tan\gamma \\ v_{\text{slide}} &= \frac{v_y}{\cos\gamma} \end{aligned}$$

Letting r be the radius of the wheel and u be the driving angular speed of the wheel, [1]

$$u = \frac{v_{drive}}{r} = \frac{1}{r}\left(v_x + v_y tan\,\gamma\right) \tag{2.4}$$

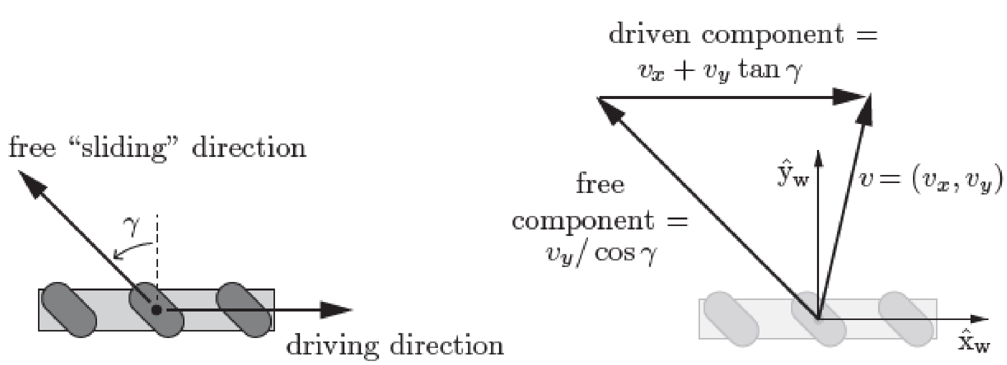

Figure 2.3: Mecanum Wheel Characteristics. [1]

Looking to Figure 2.3 where: (Left) The driving direction and the direction in which the rollers allow the wheel to slide freely. A mecanum wheel, typically have $\gamma = \pm 45°$, and (Right) The driven and free sliding speeds for the wheel velocity $v = (v_x, v_y)$ expressed in the wheel frame $\hat{x}_w - \hat{y}_w$, where the $\hat{x}_w$-axis is aligned with the forward driving direction. [1]

To derive the full transformation from the chassis velocity $\dot{q} = (\dot{\phi}, \dot{x}, \dot{y})$. to the driving angular speed u_i for wheel i, we will refer to the notation illustrated in Figure 2.4. The chassis frame $\{b\}$ is at $q = (\phi, x, y)$. in the fixed space frame $\{s\}$. The center of the wheel and its driving direction are given by (β_i, x_i, y_i) expressed in $\{b\}$, the wheel's radius is r_i, and the wheel's sliding direction is given by y_i. Then u_i is related to $\dot{q}$ by [1]

$$u_i = h_i(\phi)\dot{q} = \begin{bmatrix} \frac{1}{r_i} & \frac{\tan\gamma_i}{r_i} \end{bmatrix} \begin{bmatrix} \cos\beta_i & \sin\beta_i \\ -\sin\beta_i & \cos\beta_i \end{bmatrix} \begin{bmatrix} -y_i & 1 & 0 \\ x_i & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} \quad (2.5)$$

Reading from right to left: the first transformation expresses $\dot{q}$ as V_b; the second transformation produces the linear velocity at the wheel in $\{b\}$; the third transformation expresses this linear velocity in the wheel frame $\hat{x}_w - \hat{y}_w$; and the final transformation calculates the driving angular velocity using Equation (2.4). [1]

Evaluating Equation (2.5) for $h_i(\phi)$, we get [1]

$$h_i(\phi) = \frac{1}{r_i \cos\gamma_i} \begin{bmatrix} x_i \sin(\beta_i + \gamma_i) - y_i \cos(\beta_i + \gamma_i) \\ \cos(\beta_i + \gamma_i + \phi) \\ \sin(\beta_i + \gamma_i + \phi) \end{bmatrix}^T \quad (2.6)$$

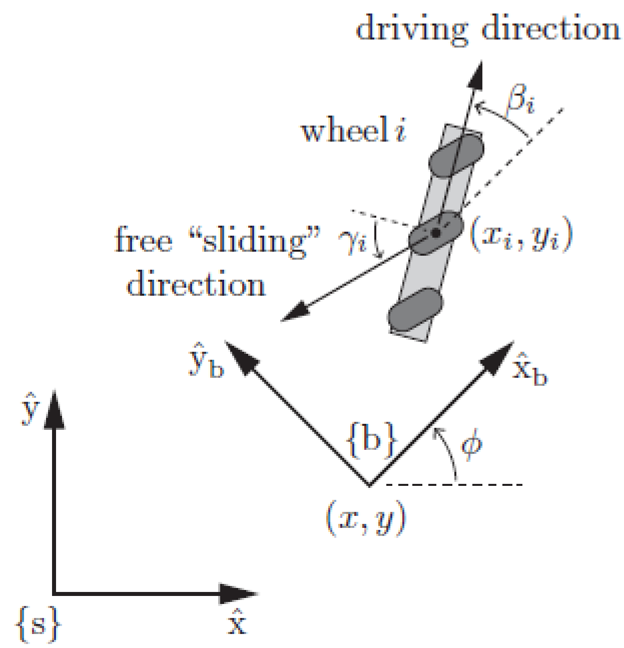

Figure 2.4: The fixed space frame $\{s\}$, a chassis frame $\{b\}$ at (ϕ, x, y) in $\{s\}$, and wheel i at (x_i, y_i) with driving direction β_i, both expressed in $\{b\}$. The sliding direction of wheel i is defined by γ_i. [1]

For an omnidirectional robot with $m \geq 3$ wheels, in our case $m = 4$, The matrix $H(\phi) \in \mathbb{R}^{m\times 3}$ mapping a desired chassis velocity $\dot{q} \in \mathbb{R}^3$ to the vector of wheel driving speeds $u \in \mathbb{R}^m$ is constructed by stacking the four rows $h_i(\phi)$: [1]

$$u = H(\phi)\dot{q} = \begin{bmatrix} h_1(\phi) \\ h_2(\phi) \\ h_3(\phi) \\ h_4(\phi) \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{x} \\ \dot{y} \end{bmatrix} \tag{2.7}$$

We can also express the relationship between u and the body twist v_b. This mapping does not depend on the chassis orientation ϕ: [1]

$$u = H(0)\mathcal{V}_b = \begin{bmatrix} h_1(0) \\ h_2(0) \\ h_3(0) \\ h_4(0) \end{bmatrix} \begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix} \tag{2.8}$$

The wheel positions and headings (β_i, x_i, y_i) in $\{b\}$, and their free sliding directions γ_i, must be chosen so that $H(0)$ is rank 3. For example, if we constructed a mobile robot of omniwheels whose driving directions and sliding directions were all aligned, the rank of $H(0)$ would be 2, and there would be no way to controllably generate translational motion in the sliding direction. [1]

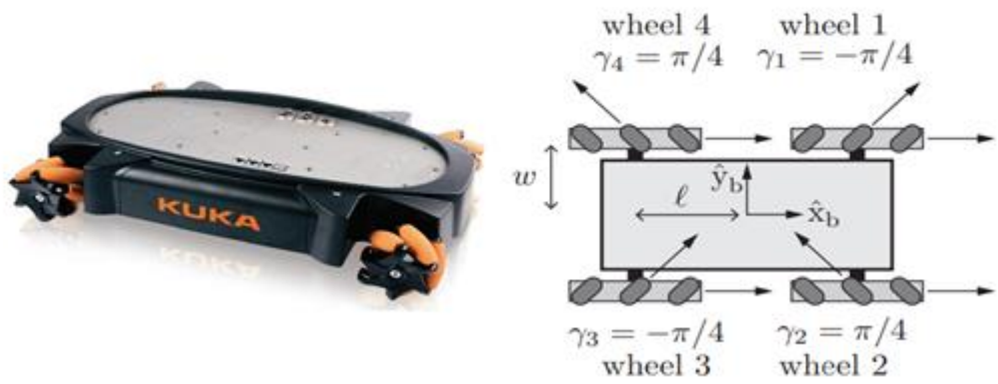

Figure 2.5: Kinematic model for the four mecanum wheels mobile robot. The radius of all wheels is r and the driving direction for each of the mecanum wheels is $\beta_i = 0$. [1]

Using the notation in Figure 2.5, the kinematic model of the mobile robot with four mecanum wheels is [1]

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = H(0)\mathcal{V}_b = \frac{1}{r}\begin{bmatrix} -\ell - w & 1 & -1 \\ \ell + w & 1 & 1 \\ \ell + w & 1 & -1 \\ -\ell - w & 1 & 1 \end{bmatrix}\begin{bmatrix} \omega_{bz} \\ v_{bx} \\ v_{by} \end{bmatrix} \tag{2.9}$$

For the mecanum robot, to move in the direction $+\hat{x}_b$, all wheels drive forward at the same speed; to move in the direction $+\hat{y}_b$, wheels 1 and 3 drive backward and wheels 2 and 4 drive forward at the same speed; and to rotate in the counterclockwise direction, wheels 1 and 4 drive backward and wheels 2 and 3 drive forward at the same speed. Note that the robot chassis is capable of the same speeds in the forward and sideways directions. [1]

1.3.2 Arm Modeling

1.3.2.1 Definition of a manipulator robot

Manipulator robot is an electronically controlled mechanism, consisting of multiple segments, that performs tasks by interacting with its environment.[7] They are also commonly referred to as robotic arms. Manipulator robots are extensively used in the industrial manufacturing sector and also have many other specialized applications. [7]

Figure 2.6**:** manipulator robot

1.3.2.2 Forward Kinematics

Forward kinematics is the use of a robot's kinematic equations to calculate the location of the end-effector given from provided joint parameter values. As a result, finding the map that transfers the joint angles to the end-effector location in Cartesian space, for an open chain, revolute joint $(5R)$ arm in our example, is primarily a "difficult" geometric issue and secondly an algebraic challenge. [3]

There are, however, methodical methods to this issue. The Denavit-Hartenberg (D-H) convention is one of the most used procedures in the field of robotics. In general, an open chain manipulator is made up of $n+1$ links linked by n joints. The 0^{th} connection is normally linked to the ground or, in our case, the manipulator base. A homogeneous transformation might represent the location and orientation of a coordinate frame on a link in relation to the previous/next one since a joint joins two successive links. [3] The D-H convention was created to make it easier to derive these homogeneous transformations and iteratively obtain the forward kinematics map by matrix multiplication of these transformations. The D-H convention, in particular, uses four parameters to fully specify the position of frame i in relation to frame $i-1$: [3]

- a_i, the distance between the origins of the two coordinate frames O_i, $O_{i'}$.
- d_i, the coordinate of O_i along z_{i-1}.
- α_i, the angle between axes z_{i-1} and z_i about axis x_i to be taken positive when rotation is made counter-clockwise.
- θ_i, the angle between axes x_{i-1} and x_i about axis z_{i-1} to be taken positive when rotation is made counter-clockwise.

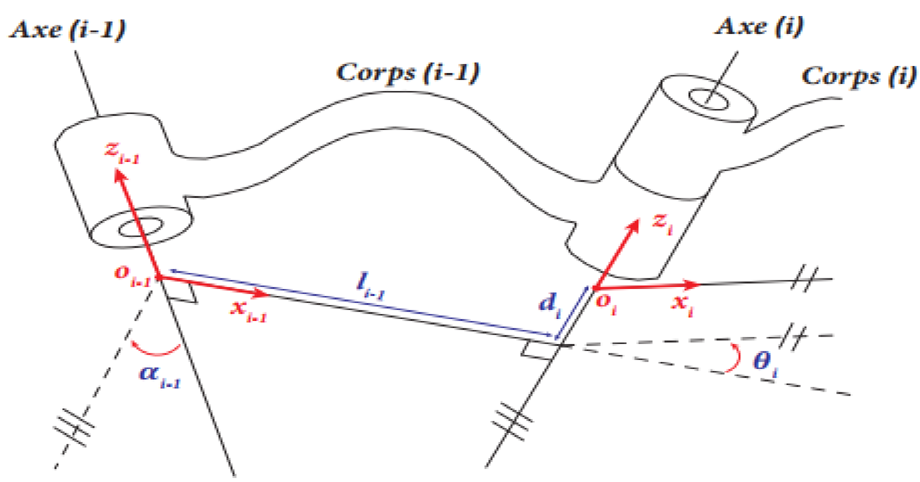

Figure 2.7: The classic Denavit-Hartenberg convention, frames and parameters.

The frames for the KUKA youBot are provided here, following the D-H regulations for frame attachment. [3]

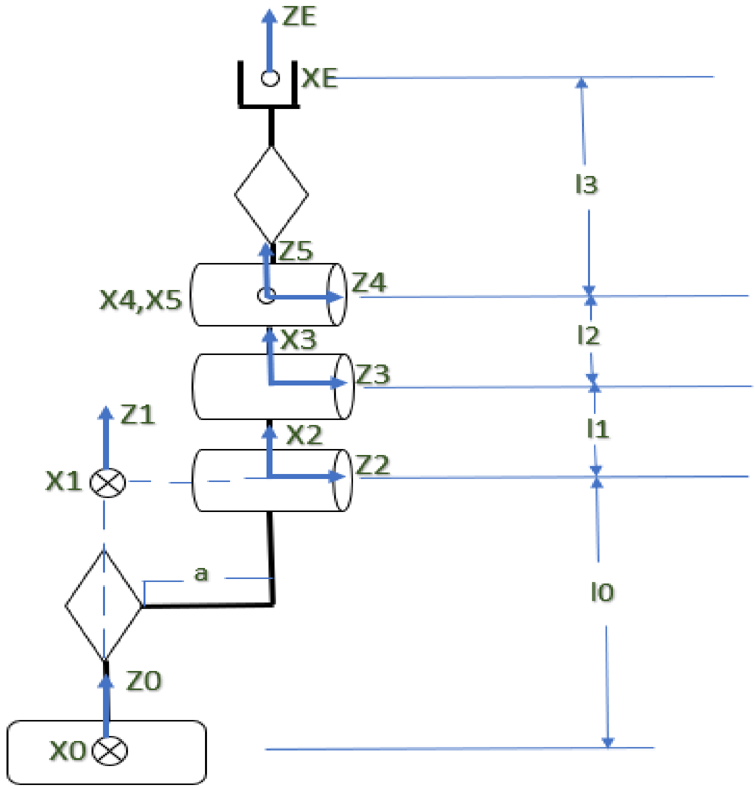

Figure 2.8: Frame attachment on the KUKA youBot.

The D-H parameters' values are listed in the table below: [3]

i	a_{i-1}	a_{i-1}	d_i	q_i
1	0	0	l_0	q_1
2	a	$\frac{\pi}{2}$	0	$q_2+\frac{\pi}{2}$
3	l_1	0	0	q_3
4	l_2	0	0	$q_4+\frac{\pi}{2}$
5	0	$\frac{\pi}{2}$	0	q_5
E	0	0	l_3	0

Table 2.1: DH parameters

Where ' a ' is the distance between the arm's two vertical z-axes. [3]

The homogeneous transformation from frame i to frame $i-1$, according to the modified D-H convention, is: [3]

$$ {}^{i-1}\mathbf{T}_i = \begin{bmatrix} cq_i & -sq_i & 0 & a_{i-1} \\ sq_i c\alpha_{i-1} & cq_i c\alpha_{i-1} & -s\alpha_{i-1} & -d_i s\alpha_{i-1} \\ sq_i s\alpha_{i-1} & cq_i s\alpha_{i-1} & c\alpha_{i-1} & d_i c\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10} $$

The end effector-to-base transformation for the KUKA youBot 5 D.O.F manipulator is: [3]

$$ {}^{0}\mathbf{T}_E = {}^{0}\mathbf{T}_1\, {}^{1}\mathbf{T}_2\, {}^{2}\mathbf{T}_3\, {}^{3}\mathbf{T}_4\, {}^{4}\mathbf{T}_5\, {}^{5}\mathbf{T}_E \tag{2.11} $$

It's worth noting that the last homogeneous transformation is essentially a translation at this point. In addition, the frame attachments were tweaked to make it easier to alter the generated equations in the latter stages of the study. [3]

We get the following frame-to-frame homogeneous transformations after substituting the parameters from the table: [3]

$$ {}^{0}\mathbf{T}_1 = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & l_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.12} $$

$$ {}^{1}\mathbf{T}_2 = \begin{bmatrix} -s_2 & -c_2 & 0 & a \\ 0 & 0 & -1 & 0 \\ c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13} $$

$$ {}^{2}\mathbf{T}_3 = \begin{bmatrix} c_3 & -s_3 & 0 & l_1 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.14} $$

$$ {}^{3}\mathbf{T}_4 = \begin{bmatrix} -s_4 & -c_4 & 0 & l_2 \\ c_4 & -s_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.15} $$

$$ {}^{4}\mathbf{T}_5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.16} $$

$$ {}^{5}\mathbf{T}_E = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.17} $$

The end-effector-to-base transition, is: [3]

$$ {}^{0}\mathbf{T}_E = \begin{bmatrix} -c_1 c_{234} c_5 + s_1 s_5 & c_1 c_{234} s_5 + c_5 s_1 & -c_1 s_{234} & c_1(a - l_1 s_2 - l_2 s_{23} - l_3 s_{234}) \\ -s_1 c_{234} c_5 - c_1 s_5 & s_1 c_{234} s_5 - c_1 c_5 & -s_1 s_{234} & s_1(a - l_1 s_2 - l_2 s_{23} - l_3 s_{234}) \\ -s_{234} c_5 & s_{234} s_5 & c_{234} & l_0 + l_1 c_2 + l_2 c_{23} + l_3 c_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.18} $$

A few last thoughts about kinematics. Some methods can assess the correctness of the forward kinematics map, at least in part. For starters, because the homogeneous transformation's orientation portion, i.e., the top left 3×3 block matrix, is an orthogonal matrix, the squared sum of its lines and columns must equal $+1$. This feature is comparable to having a $+1$ determinant. In terms of the length's component, we may replace the joint variables with values that reflect a known position for the end-effector (e.g., home position, full stretch, etc.) and check if the map holds up. For our transition, both prerequisites are satisfied. [3]

1.3.2.3 Dynamic model – Arm

We have calculated our dynamic model using MATLAB and we have written the full results in the appendix due to the big size of it.

Let the Euler-Lagrange equation: [11]

$$ \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \tag{2.19} $$

And [11]

$$ \mathcal{L} = \mathcal{T} - \mathcal{U} \tag{2.20} $$

Where: [11]

$\mathcal{T}$: Kinetic energy of the system.

$\mathcal{U}$: Potential energy of the system.

q_i : $-^{\text{th}}$ generalized system coordinate.

τ_i : Generalized force applied to the 1st element.

And the equation that describes the dynamics of a manipulator robot: [11]

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau \tag{2.21}$$

Where: [11]

- $\boldsymbol{M(q)}$**:** is the inertia matrix.
- $\boldsymbol{C(q,\dot{q})}$**:** is the Coriolis forces matrix.
- $\boldsymbol{G(q)}$**:** is the gravitational forces vector.

$$M(q) = \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} \\ M_{21} & M_{22} & M_{23} & M_{24} & M_{25} \\ M_{31} & M_{32} & M_{33} & M_{34} & M_{35} \\ M_{41} & M_{42} & M_{43} & M_{44} & M_{45} \\ M_{51} & M_{52} & M_{53} & M_{54} & M_{55} \end{bmatrix} \tag{2.22}$$

If we put [11]

$$C_0(q,\dot{q}) = C(q,\dot{q})\dot{q} = \dot{M}(q)\dot{q} + \frac{1}{2}\frac{\partial}{\partial q}[\dot{q}^T M(q)\dot{q}] \tag{2.23}$$

We will get [11]

$$C(q,\dot{q}) = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} \end{bmatrix} \tag{2.24}$$

And if we put [11]

$$G(q) = \frac{\partial \mathcal{U}(q)}{\partial q} \tag{2.25}$$

We will have [11]

$$G(q) = \begin{bmatrix} G_{11} \\ G_{21} \\ G_{31} \\ G_{41} \\ G_{51} \end{bmatrix} \tag{2.26}$$

Note: The elements of the matrixes $\boldsymbol{M(q)}$, $\boldsymbol{C(q,\dot{q})}$ and $\boldsymbol{G(q)}$ can be found in **appendix (1)**.

1.3.3 Combined System Kinematics

The kinematic and dynamic models for the manipulator and the mobile platform were developed and studied independently in the previous chapter. In this chapter, the arm is assumed to be installed on the platform, and the kinematic model (forward and differential kinematics) is generated. [3]

1.3.3.1 Kinematics Forward

To begin, the following relationship describes the homogeneous transition from the end-effector to the global inertial frame: [3]

$$^{I}\mathbf{T}_E = \begin{bmatrix} ^{I}\mathbf{R}_0 \; ^{0}\mathbf{R}_E & \mathbf{r} + \, ^{I}\mathbf{R}_0 \, ^{0}\mathbf{r}_E \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{2.27}$$

Where:

The vector $^{0}\mathbf{r}_E$ is the first three elements of the fourth column of (2.18) To be precise: [3]

$$^{I}\mathrm{R}_E = \, ^{I}\mathrm{R}_0^0\mathrm{R}_E = \begin{bmatrix} c_{234}c_5\cos(q_1-\theta_P) - s_5\sin(q_1-\theta_P) & -c_{234}\cos(q_1-\theta_P)s_5 - c_5\sin(q_1-\theta_P) & \cos(q_1-\theta_P)s_{234} \\ \cos(q_1-\theta_P)s_5 + c_{234}c_5\sin(q_1-\theta_P) & -c_{234}\sin(q_1-\theta_P)s_5 + c_5\cos(q_1-\theta_P) & s_{234}\sin(q_1-\theta_P) \\ -s_{234}c_5 & s_{234}s_5 & c_{234} \end{bmatrix} \tag{2.28}$$

And [3]

$$\mathrm{r} + \, ^{I}\mathrm{R}_0 \, ^{0}\mathrm{r}_E = \begin{bmatrix} x_P + \, ^{P}x_0 - \cos(q_1-\theta_P)(a - l_1s_2 - l_2s_{23} - l_3s_{234}) \\ y_P + \, ^{P}y_0 - \sin(q_1-\theta_P)(a - l_1s_2 - l_2s_{23} - l_3s_{234}) \\ ^{I}z_0 + l_0 + l_1c_2 + l_2c_{23} + l_3c_{234} \end{bmatrix} \tag{2.29}$$

1.3.3.2 Differential Kinematics

Now we must compute the composite Jacobian J^*. However, the geometric technique used in the previous chapter for the manipulator is no longer valid Thus, the time differentiation method, i.e., differentiating the relationship (2.27) w.r.t. time in order to derive relationships between end-effector and joint velocities/control variables, will be implemented in this case. [3]

The Jacobian contains two portions, one for the end-linear effector's speeds and the other for its rotational velocities, as is customary. We'll start with the linear section, J_L^*. When the first three components in the fourth column of (2.27) are differentiated (in terms of time), we get: [3]

$$\frac{d}{dt}(\mathrm{r} + {}^{I}\mathrm{R}_0^0\mathrm{r}_E) = \frac{d\mathrm{r}}{dt} + \frac{d({}^{I}\mathrm{R}_0)}{dt}\,{}^{0}\mathrm{r}_E + {}^{I}\mathrm{R}_0\frac{d({}^{0}\mathrm{r}_E)}{dt} \tag{2.30}$$

For the first term we have: [3]

$$\frac{d\mathbf{r}}{dt} = \frac{d}{dt}[x_P + {}^{P}x_0 \quad y_P + {}^{P}y_0 \; {}^{I}z_0]^T = [\dot{x}_P \quad \dot{y}_P \quad 0]^T \tag{2.31}$$

In the second term, the time derivative of a rotation matrix appears. As we know, for any rotation matrix R, its time derivative is $\dot{\mathbf{R}} = (\boldsymbol{\omega})^x\mathbf{R}$ where $(\boldsymbol{\omega})^x$ is the skew symmetric matrix operator, as defined in Chapter 4, applied on the angular velocity of the rotating body. In our specific case $\boldsymbol{\omega}_P = [0 \quad 0 \quad \dot{\theta}_P]^T$. Hence: [3]

$$\frac{d({}^{I}R_0)^0}{dt}\boldsymbol{r}_E = (\boldsymbol{\omega}_P)^x\;{}^{I}R_0\,{}^{0}\boldsymbol{r}_E = \dot{\theta}_P\begin{bmatrix} -sin\,\theta_P & cos\,\theta_P & 0 \\ -cos\,\theta_P & sin\,\theta_P & 0 \\ 0 & 0 & 0 \end{bmatrix}{}^{0}\boldsymbol{r}_E \tag{2.32}$$

Lastly, for the third term we have: [3]

$${}^{I}\mathrm{R}_0\frac{d({}^{0}\mathrm{r}_E)}{dt} = {}^{I}\mathrm{R}_0\frac{\partial\,{}^{0}\mathrm{r}_E}{\partial \mathrm{q}_m}\frac{\partial \mathrm{q}_m}{\partial t} = {}^{I}\mathrm{R}_0\,{}^{0}\mathrm{J}_{Gm,L}\dot{\mathrm{q}}_m \tag{2.33}$$

Where: ${}^{0}\mathbf{J}_{Gm,L}$ is the part of the manipulator geometric Jacobian that corresponds to the linear velocities, i.e., the first three rows. Having said the above, (2.30) becomes: [3]

$$\frac{d}{dt}(\mathrm{r} + {}^{I}\mathrm{R}_0\,{}^{0}\mathrm{r}_E) = [\dot{x}_P\dot{y}_P 0]^T + \dot{\theta}_P\begin{bmatrix} -\sin\theta_P & \cos\theta_P & 0 \\ -\cos\theta_P & \sin\theta_P & 0 \\ 0 & 0 & 0 \end{bmatrix}{}^{0}\mathrm{r}_E + {}^{I}\mathrm{R}_0\,{}^{0}\,\mathrm{J}_{Gm,L}\dot{\mathrm{q}}_m \tag{2.34}$$

Where: [3]

$$\dot{\theta}_P\begin{bmatrix} -\sin\theta_P & \cos\theta_P & 0 \\ -\cos\theta_P & \sin\theta_P & 0 \\ 0 & 0 & 0 \end{bmatrix}{}^{0}\mathbf{r}_E = \begin{bmatrix} \sin(q_1 - \theta_P)(a - l_1s_2 - l_2s_{23} - l_3s_{234}) \\ -\cos(q_1 + \theta_P)(a - l_1s_2 - l_2s_{23} - l_3s_{234}) \\ 0 \end{bmatrix}\dot{\theta}_P \tag{2.35}$$

And: [3]

$$^{I}\mathbf{R}_0\,{}^{0}\mathbf{J}_{m,L} = \begin{bmatrix} \sin(q_1-\theta_P)(a-l_1s_2-l_2s_{23}-l_3s_{234}) & \cos(q_1-\theta_P)(l_1c_2+l_2c_{23}+l_3c_{234}) & \cos(q_1-\theta_P)(l_2c_{23}+l_3c_{234}) & l_3c_{234}\cos(q_1-\theta_P) & 0 \\ -\cos(q_1-\theta_p)(a-l_1s_2-l_2s_{23}-l_3s_{234}) & \sin(q_1-\theta_p)(l_1c_2+l_2c_{23}+l_3c_{234}) & \sin(q_1-\theta_p)(l_2c_{23}+l_3c_{234}) & l_3c_{234}\sin(q_1-\theta_p) & 0 \\ 0 & -l_1s_2-l_2s_{23}-l_3s_{234} & -l_2s_{23}-l_3s_{234} & -l_3s_{234} & 0 \end{bmatrix} \tag{2.36}$$

The angular part $\mathbf{J}_A^*$ is a bit more difficult to calculate. For this particular purpose we employ a specific representation of the angular velocities of the end-effector. To be precise, we cannot construct this part of the Jacobian geometrically and the use of a representation other than the vector $[\omega_x\ \omega_y\ \omega_z]$ will enable us to obtain this part by differentiation w.r.t. time. As a first approach we use the Roll-Pitch-Yaw (RPY) Euler angles. If we begin from the complete rotation matrix $^{\mathrm{I}}\mathrm{R}_{\mathrm{E}}$ and its elements r_{ij} we have: [3]

$$\alpha_E = \text{Atan 2}(r_{21}, r_{11}) = f_\alpha(\mathrm{q}) \tag{2.37}$$

$$\beta_E = \text{Atan 2}\left(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}\right) = f_\beta(\mathrm{q}) \tag{2.38}$$

$$\gamma_E = \text{Atan 2}(r_{32}, r_{33}) = f_\gamma(\mathrm{q}) \tag{2.39}$$

Note that the above solution for β_E is in the range $(-\frac{\pi}{2};\frac{\pi}{2})$ and all solutions degenerate when $\cos \beta_{\mathrm{E}} = 0$. Then, only the sum or difference of α_E; γ_E can be calculated. Regardless, we assume for the time being that this is not the case and $\beta_{\mathrm{E}} \neq 90^\circ$. The angular part of the Jacobian is extracted then by the below relationships: [3]

$$\dot{\alpha}_E = \frac{\partial f_\alpha}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = \frac{\partial f_\alpha}{\partial q_1}\dot{q}_1 + \cdots + \frac{\partial f_\alpha}{\partial q_n}\dot{q}_n \tag{2.40}$$

$$\dot{\beta}_E = \frac{\partial f_\beta}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = \frac{\partial f_\beta}{\partial q_1}\dot{q}_1 + \cdots + \frac{\partial f_\beta}{\partial q_n}\dot{q}_n \tag{2.41}$$

$$\dot{\gamma}_E = \frac{\partial f_\gamma}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = \frac{\partial f_\gamma}{\partial q_1}\dot{q}_1 + \cdots + \frac{\partial f_\gamma}{\partial q_n}\dot{q}_n \tag{2.42}$$

Where the partial derivatives constitute the elements of $\mathbf{J}_A^*$. That said, we have: [3]

$$\dot{\alpha}_E = \dot{q}_1 + \frac{s_{234}s_5c_5}{1-s_{234}^2c_5^2}\dot{q}_2 + \frac{s_{234}s_5c_5}{1-s_{234}^2c_5^2}\dot{q}_3 + \frac{s_{234}s_5c_5}{1-s_{234}^2c_5^2}\dot{q}_4 + \frac{c_{234}}{1-s_{234}^2c_5^2}\dot{q}_5 - \dot{\theta}_P \tag{2.43}$$

$$\dot{\beta}_E = -\frac{c_{234}c_5}{\sqrt{1-s_{234}^2c_5^2}}\dot{q}_2 - \frac{c_{234}c_5}{\sqrt{1-s_{234}^2c_5^2}}\dot{q}_3 - \frac{c_{234}c_5}{\sqrt{1-s_{234}^2c_5^2}}\dot{q}_4 + \frac{s_{234}s_5}{\sqrt{1-s_{234}^2c_5^2}}\dot{q}_5 \tag{2.44}$$

$$\dot{\gamma}_E = \frac{s_5}{1-s_{234}^2 c_5^2}\dot{q}_2 + \frac{s_5}{1-s_{234}^2 c_5}\dot{q}_3 + \frac{s_5}{1-s_{234}^2 c_5^2}\dot{q}_4 + \frac{0.5 c_5 \sin[2(q_2+q_3+q_4)]}{1-s_{234}^2 c_5^2}\dot{q}_5 \tag{2.45}$$

When replace the DOFs of the platform with the control variables $\boldsymbol{u}_{\boldsymbol{p},\mathbf{0}}$, the parts of the Jacobian become: [3]

$$\mathrm{J}_{L,0}^{*} = \begin{bmatrix} {}^{I}\mathrm{R}_0 & {}^{0}\mathrm{J}_{Gm,L}\big|_{\text{row 1}} & \kappa_1 & \kappa_2 & \kappa_1 & \kappa_2 \\ {}^{I}\mathrm{R}_0 & {}_{0}\mathrm{J}_{Gm,L}\big|_{\text{row 2}} & -\kappa_3 & \kappa_3 & \kappa_4 & -\kappa_4 \\ {}^{I}\mathrm{R}_0 & {}^{0}\mathrm{J}_{Gm,L}\big|_{\text{row 3}} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.46}$$

Where: [3]

$$\kappa_1 = \frac{r}{4} - \frac{r(a-l_1 s_2 - l_2 s_{23} - l_3 s_{234})}{4(l_x+l_y)}\sin(q_1-\theta_p) \tag{2.47}$$

$$\kappa_2 = \frac{r}{4} + \frac{r(a-l_1 s_2 - l_2 s_{23} - l_3 s_{234})}{4(l_x+l_y)}\sin(q_1-\theta_p) \tag{2.48}$$

$$\kappa_3 = \frac{r}{4} + \frac{r(a-l_1 s_2 - l_2 s_{23} - l_3 s_{234})}{4(l_x+l_y)}\cos(q_1+\theta_p) \tag{2.49}$$

$$\kappa_4 = \frac{r}{4} - \frac{r(a-l_1 s_2 - l_2 s_{23} - l_3 s_{234})}{4(l_x+l_y)}\cos(q_1+\theta_p) \tag{2.50}$$

$$\mathbf{J}_{A,o}^{*} = \begin{bmatrix} 1 & +\frac{s_{234}s_5c_5}{1-s_{224}^2c_5^2} & +\frac{s_{234}s_5c_5}{1-s_{234}^2c_5^2} & +\frac{s_{234}s_5c_5}{1-s_{234}^2c_5^2} & +\frac{c_{234}}{1-s_{234}^2c_5^2} & \kappa_0 & -\kappa_0 & \kappa_0 & -\kappa_0 \\ 0 & -\frac{c_{234}}{\sqrt{1-s_{23}^2c_5^2}} & -\frac{c_{234}}{\sqrt{1-s_{234}^2c_5^2}} & -\frac{c_{234}}{\sqrt{1-s_{234}^2c_5^2}} & \frac{s_{234}s_5}{\sqrt{1-s_{234}^2c_5^2}} & 0 & 0 & 0 & 0 \\ 0 & \frac{s_5}{1-s_{234}^2c_5^2} & \frac{s_5}{1-s_{234}^2c_5^2} & \frac{s_5}{1-s_{234}^2c_5^2} & \frac{0.5c_5\sin[2(q_2+q_3+q_4)]}{1-s_{234}^2c_5^2} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.51}$$

Where: [3]

$$\kappa_0 = \frac{r}{4(l_x+l_y)} \tag{2.52}$$

$$\mathrm{J}_{nh}^{*} = \begin{bmatrix} \mathrm{J}_{L,nh}^{*} \\ \mathrm{J}_{A,nh}^{*} \end{bmatrix} \tag{2.53}$$

1.4 Conclusion

At the end of this chapter, we can now say that we have some knowledge about Kuka youbot (components, characteristics and uses). Moreover, The Kuka youbot robot models were developed in an independently and combined forms in this chapter, by studying the mobile platform kinematic model. In addition, the manipulator modeling starting by the forward kinematics then the dynamics were also discussed. Then, the kinematic model (forward and differential kinematics) for our Kuka youbot robot was developed and studied for the combined system kinematic (assuming that the arm is mounted on the platform). While the next chapter will deal with control, for us it’s the important part because it will verify the correctness of the studied models.

Chapter 2 Control of Kuka youBot

2.1 Introduction

For a robot arm mounted on a movable base, movable manipulation describes the coordination of movement of the base and joints of the robot to achieve the desired motion at the end-effector. Usually, the arm's motion can be controlled more precisely than the base's motion, so the most common types of mobile manipulation involve driving the base, stopping it, and then letting the arm perform the task with precise motion. This chapter also covers a trajectory definition and a simulation part using MATLAB software to study and control our Kuka youbot by using the obtained models from the previous chapters in order to see results and compare them.

2.2 Trajectory

Trajectory planning is the generation of reference inputs to the motion control system. Goal is to move the manipulator from initial pose to final desired pose. The transition is characterized by motion laws requiring the actuators to apply joint forces that do not violate the saturation limits. It is necessary to develop planning algorithms that generate suitable smooth trajectories. [2]

2.2.1 Path and Trajectory

The minimum requirement for the manipulator is the ability to move from the initial position to the specified final position. In order to avoid confusion between terms that are often used as synonyms, the difference between path and path must be explained. The path indicates the position of points in the common space, or in the run space, that the processor must follow in the execution of the assigned motion; It is the path in which the law of timing is determined. [2]

2.2.2 Joint Space Trajectories

Joint space trajectory algorithms should have the following features: [2]

- The trajectories should not be computationally expensive.
- The joint positions and velocities should be continuous functions of time.
- The trajectories should be as smooth as possible.

We have in this study two part of trajectory: [2]

2.2.2.1 Point-to-Point Motion

Third-order polynomials can be used where the position, velocity, and accelerations are constructed as: [2]

$$q(t) = a_3t^3 + a_2t^2 + a_1t + a_0 \tag{3.1}$$

$$\dot{q}(t) = 3a_3t^2 + 2a_2t + a_1 \tag{3.2}$$

$$\ddot{q}(t) = 6a_3t + 2a_2 \tag{3.3}$$

These equations have four unknowns. Therefore, the following should be specified: initial and final positions; initial and final velocities: [2]

$$a_0 = q_i \tag{3.4}$$

$$a_1 = \dot{q}_i \tag{3.5}$$

$$a_3t_f^3 + a_2t_f^2 + a_1t_f + a_0 = q_f \tag{3.6}$$

$$3a_3t_f^2 + 2a_2t_f + a_1 = \dot{q}_f \tag{3.7}$$

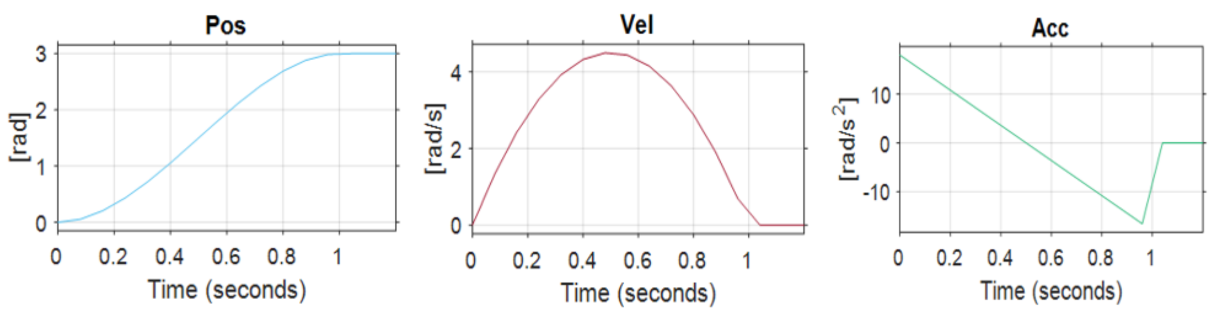

Figure 3.1: Time history of position, velocity and acceleration with a cubic polynomial timing law

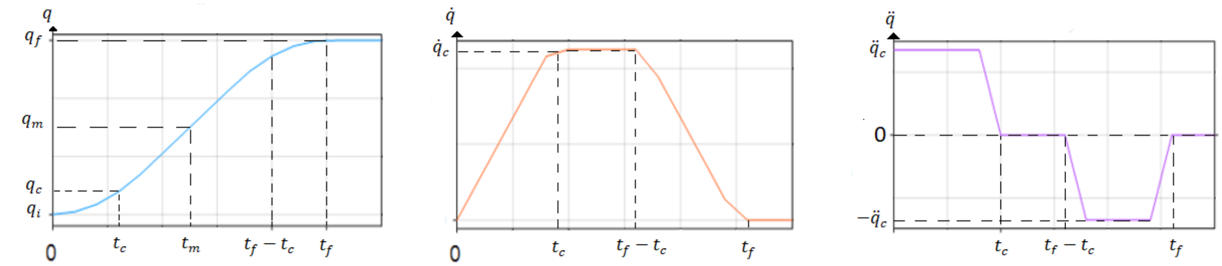

Figure 3.2: Characterization of a timing law with trapezoidal velocity profile in terms of position, velocity and acceleration

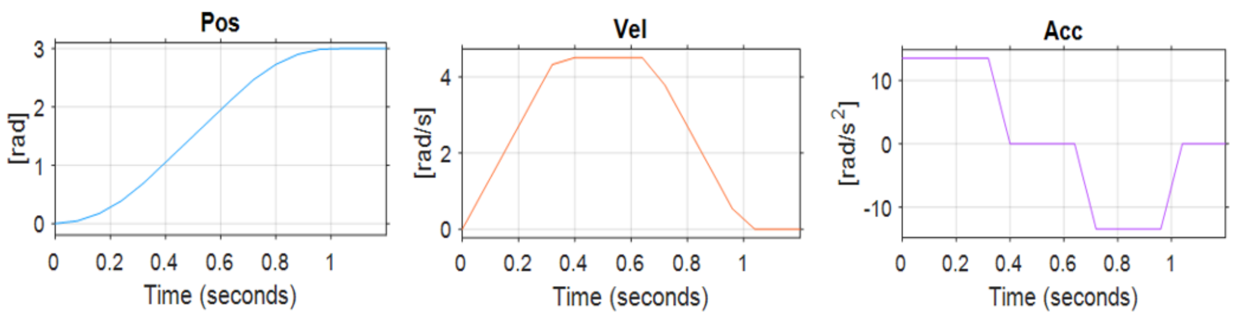

Figure 3.3: Time history of position, velocity and acceleration with a trapezoidal velocity profile timing law

2.2.3 Operational Space Trajectories

A joint space trajectory planning algorithm generates a time sequence of values for the joint variables $q(t)$ so that the manipulator is taken from the initial to the final configuration. [2]

On the other hand, the end-effector motion follows a geometrically specified path in the operational space. Then, the corresponding sequence of values for the joint variables is computed using inverse kinematics algorithm. [2]

2.2.3.1 Path Primitives

Let p be a (3×1) vector, then we can define a continuous vector function $f(s)$. then, $p = f(s)$ Where s is the arc length. This equation defines the parametric representation of a path G. [2]

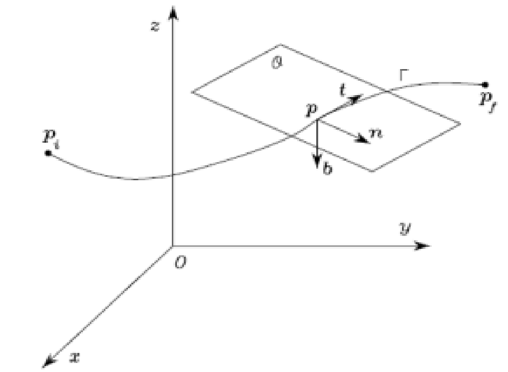

Figure 3.4 Path Primitives

We have two types of paths in operational trajectory: [2]

a) Rectilinear Path

We have [2]

$$p(s) = p_i + \frac{s}{\|p_f - p_i\|}(p_f - p_i) \tag{3.8}$$

$$\frac{dp}{ds} = \frac{1}{\|p_f - p_i\|}(p_f - p_i) \tag{3.9}$$

$$\frac{d^2p}{ds^2} = 0 \tag{3.10}$$

b) Circular Path

$$p(s) = c + Rp'(s) \tag{3.11}$$

$$\frac{dp}{ds} = R\begin{bmatrix} -\sin\left(\frac{s}{\rho}\right) \\ \cos\left(\frac{s}{\rho}\right) \\ 0 \end{bmatrix} \tag{3.12}$$

$$\frac{d^2p}{ds^2} = R\begin{bmatrix} -\frac{\cos\left(\frac{s}{\rho}\right)}{\rho} \\ -\frac{\sin\left(\frac{s}{\rho}\right)}{\rho} \\ 0 \end{bmatrix} \tag{3.13}$$

Where: [2]

$$p'(s) = \begin{bmatrix} \rho\cos(s/\rho) \\ \rho\sin(s/\rho) \\ 0 \end{bmatrix} \tag{3.14}$$

$$\rho = \|p_i - c\| \tag{3.15}$$

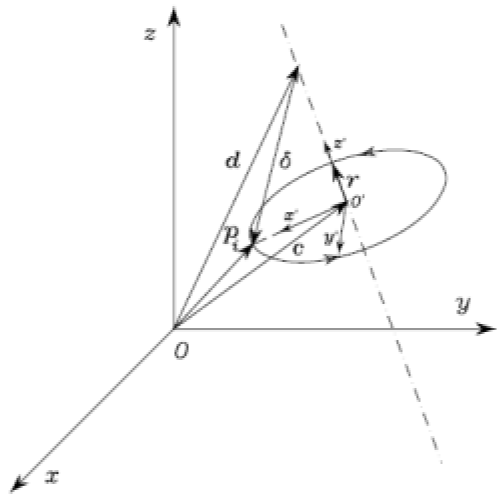

Figure 3.5: Circular Path

2.2.3.2 Position

Let x_e be the vector of operational space variables expression the pose of the manipulator's end effector. Generating a trajectory in the operational space means to determine a function $x_e\ (t)$ taking the end-effector frame from the initial to the final pose in time tf: [2]

$$\boldsymbol{p}_e = \boldsymbol{p}_0 + \sum_{j=1}^{N} \frac{s_j}{\|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|}(\boldsymbol{p}_j - \boldsymbol{p}_{j-1}) \tag{3.16}$$

$$\dot{p}_e = \sum_{j=1}^{N} \frac{\dot{s}_j}{\|p_j - p_{j-1}\|}(p_j - p_{j-1}) = \sum_{j=1}^{N} \dot{s}_j t_j \tag{3.17}$$

$$\ddot{p}_e = \sum_{j=1}^{N} \frac{\ddot{s}_j}{\|p_j - p_{j-1}\|}(p_j - p_{j-1}) = \sum_{j=1}^{N} \ddot{s}_j t_j \tag{3.18}$$

2.2.3.3 Orientation [2]

$$\phi_e = \phi_i + \frac{s}{\|\phi_f - \phi_i\|}(\phi_f - \phi_i) \tag{3.19}$$

$$\dot{\phi}_e = \frac{\dot{s}}{\|\phi_f - \phi_i\|}(\phi_f - \phi_i) \tag{3.20}$$

$$\ddot{\phi}_e = \frac{\ddot{s}}{\|\phi_f - \phi_i\|}(\phi_f - \phi_i) \tag{3.21}$$

2.3 Simulation

In this section, we have used MATLAB software for our simulation. In addition, our work here is devised into two main parts, the first part is a separated system that control the robot platform first then after it reaches the desired position, at that time the control of the manipulator will start. Moreover, the second part is the control of the combined system.

2.3.1 Separated system

As we mention before in the separated system section, we will control the robot platform first then after it reaches the desired position, at that time an error condition block will check if the platform mobile is in the right location by verifying the position error ($err \leq 10^{-4}$), if it is true, it will give the start signal to the manipulator control. The figure below shows the diagram block of this control process:

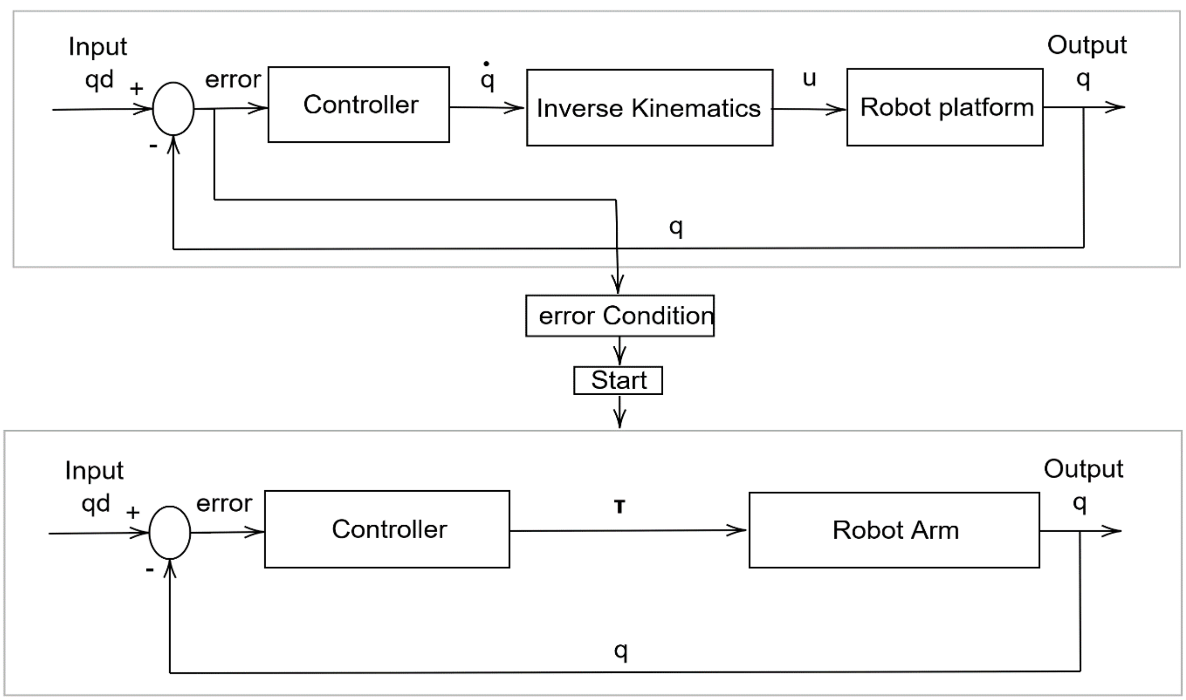

Figure 3.6: Diagram block of the separated system control process.

2.3.1.1 Mobile platform

The control process:

We will use the feedback control by Given a desired trajectory $q_d(t)$, we can adopt the feedforward plus PI feedback controller to track the trajectory: [1]

$$\dot{q}(t) = \dot{q}_d(t) + K_p(q_d(t) - q(t)) + K_i\int_0^t (q_d(\mathrm{t}) - q(\mathrm{t}))dt \tag{3.22}$$

Where $K_p = k_p I \in \mathbb{R}^{3\times 3}$ and $K_i = k_i I \in \mathbb{R}^{3\times 3}$ [1] have positive values along the diagonal and $q(t)$ is an estimate of the actual configuration derived from sensors (pose). Then $\dot{q}(t)$ can be converted to the commanded wheel driving velocities $u(t)$ using Equation (2.7). After that, the final obtained wheels speeds $u = [u_1\ u_2\ u_3\ u_4]^T$ will command and drive the mobile platform to the desired waypoints. The structure of this control process is showed in the figure below:

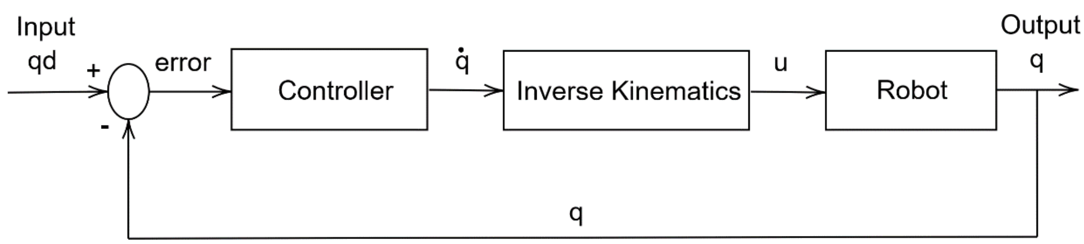

Figure 3.7: Diagram block of Feedback control of the mobile platform (Simulink).

We gave our platform robot the following desired waypoints:

- **Point 1:** $[p_x = 0\ ; p_y = 0.5\ ;\ \theta = 0]$.
- **Point 2:** $[p_x = 1\ ; p_y = 0.5\ ;\ \theta = 0]$.
- **Point 3:** $[p_x = 1\ ; p_y = 1\ ;\ \theta = 0.5]$.

Where:

$\boldsymbol{p_x}$**:** is the position along the axe x.

$\boldsymbol{p_y}$**:** is the position along the axe y.

$\boldsymbol{\theta}$**:** is the rotation angle along the axe z.

a. Trajectory tracking

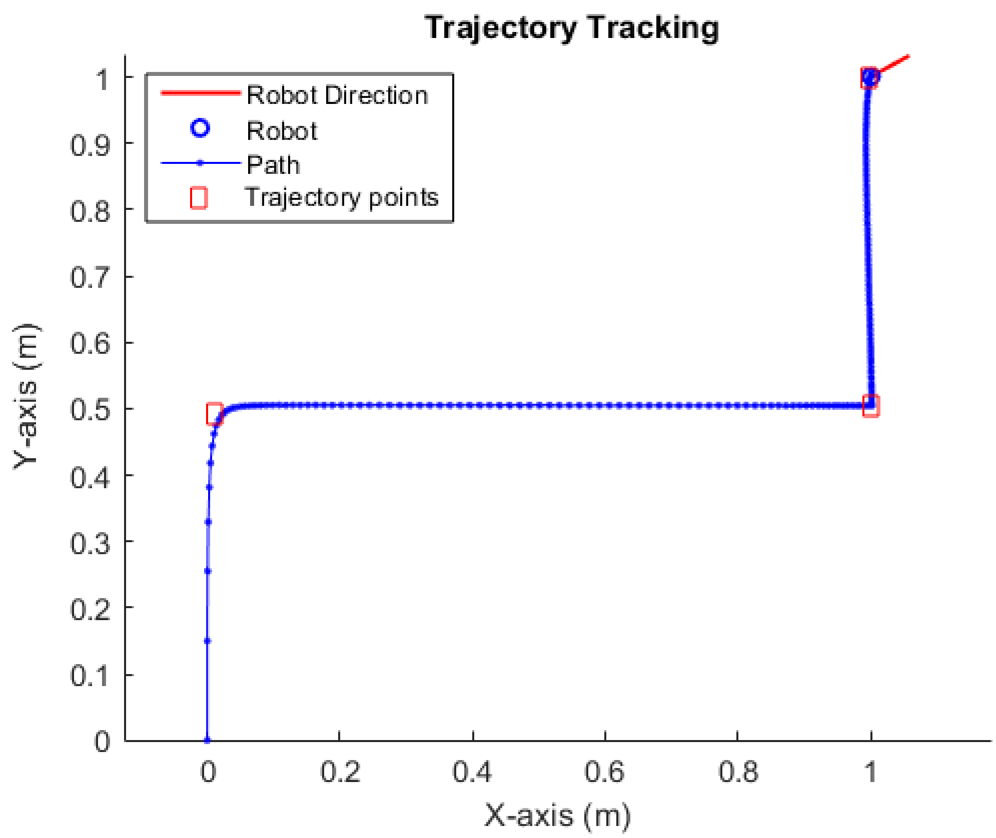

Figure 3.8: Trajectory tracking by the robot platform mobile.

From the figure, we can clearly see that our robot platform is following the trajectory (path) perfectly after passing through the first point then the second one ending at the final point with the exact given coordination and exact rotation angle, which means that our controller and our platform model are correct and reliable.

b. Position tracking response:

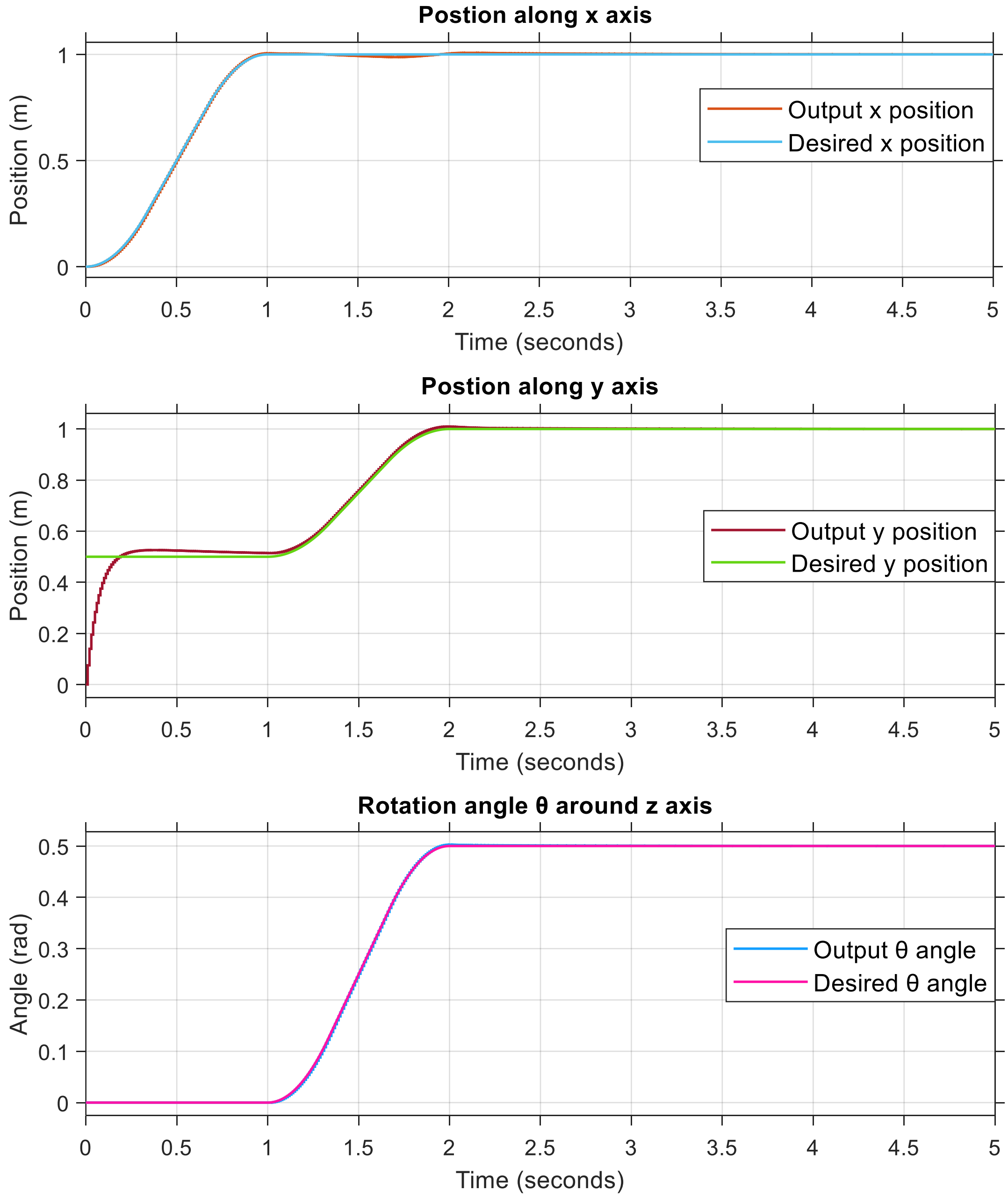

Figure 3.9: the desired and output position and orientation results by the robot platform mobile.

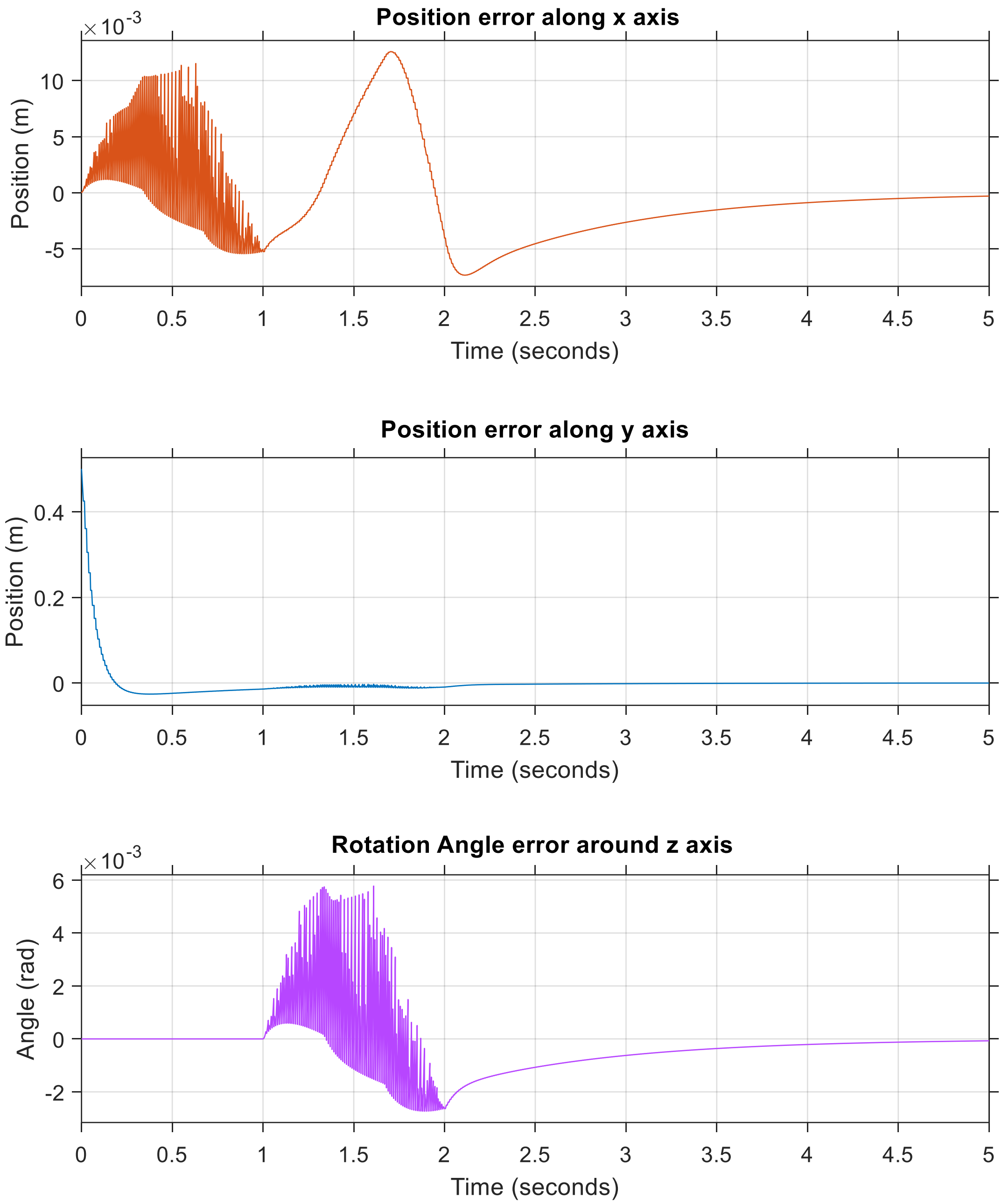

Figure 3.10: Trajectory tracking error.

From these figures, we can see that the response to the desired given waypoints and given rotation angles by the robot platform is very smooth, reasonable and accurate, that is what makes our PI controller reliable.

c. Wheels speeds Commands:

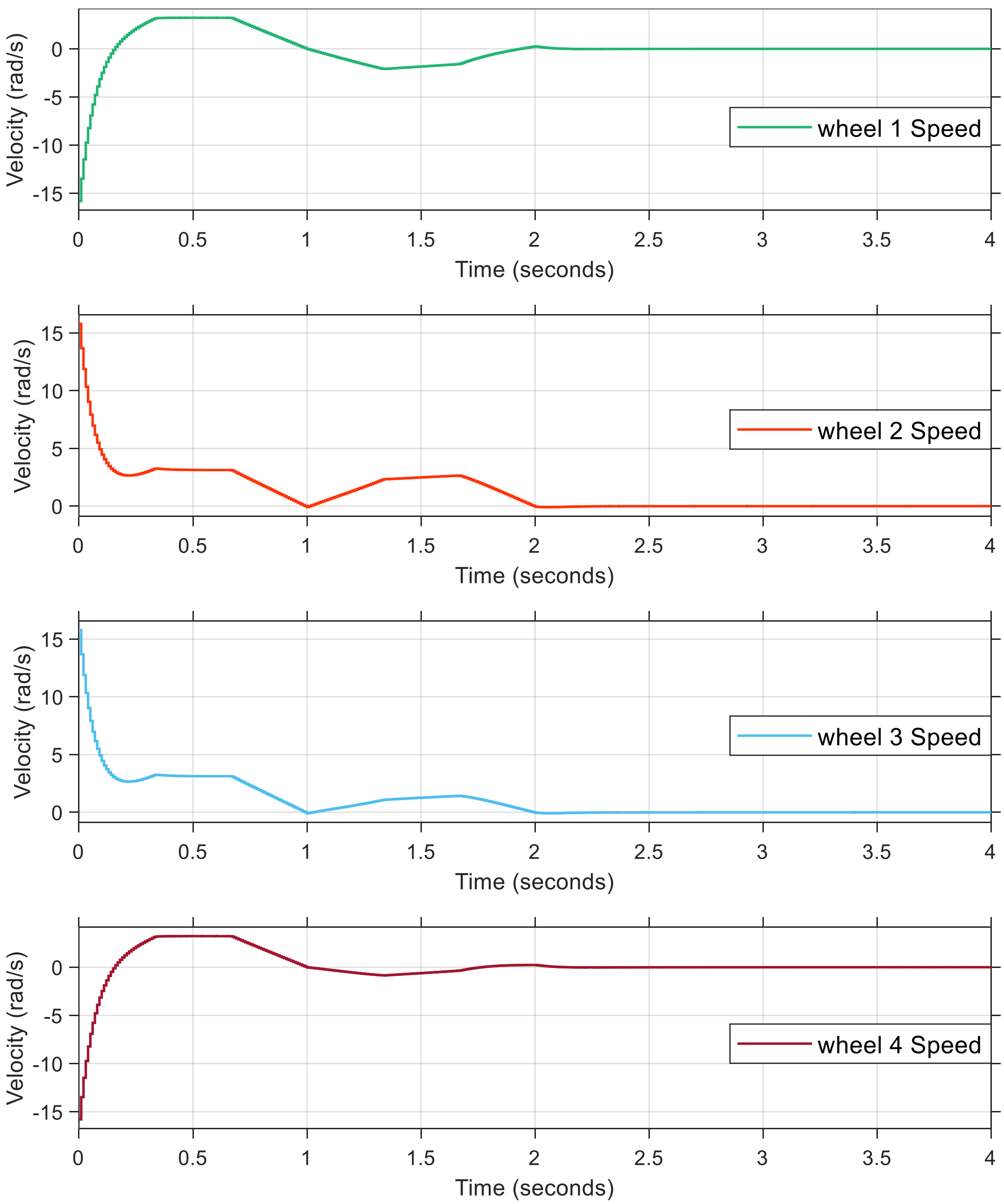

Figure 3.11: The Giving Wheels speeds Commands to the mobile platform.

2.3.1.2 Manipulator

In this section, we will see the simulation results of the manipulator part in three different cases, the first one is by using a PD controller, the second one is by using a gravity compensation controller, the last one is by using a computed torques controller. While the simulation, we will notice that our manipulator starts moving after 6.2 seconds that is because it was waiting for the robot platform to reach the desired position and then he starts moving.

We gave our manipulator joints the following desired angles:

- joint 1: [$\theta = 0.5$].
- joint 2: [$\theta = 0.3$].
- joint 3: [$\theta = 0.2$].
- joint 4: [$\theta = 0.7$].
- joint 5: [$\theta = 0.1$].

A. PD controller results

The control process:

The control process is going to be in the joint space. We will use the derivative proportional PD controller by using the feedback of both actual joints' angles (actual trajectory) $q(t)$ and the actual joints speeds $\dot{q}(t)$. The control law is given as follows: [11]

$$\tau = K_p e + K_d \dot{e} \tag{3.23}$$

$$e = q_d - q$$

With $K_p \in \mathbb{R}^{5\times5}$, $K_d \in \mathbb{R}^{5\times5}$ are defined positive matrices [11], q_d is the trajectory reference (desired). After that, the final obtained joints torque $\tau = [\tau_1\ \tau_2\ \tau_3\ \tau_4\ \tau_5]^T$ will command and drive the manipulator joints to the desired trajectory. The structure of this control process is showed in the figure below: [11]

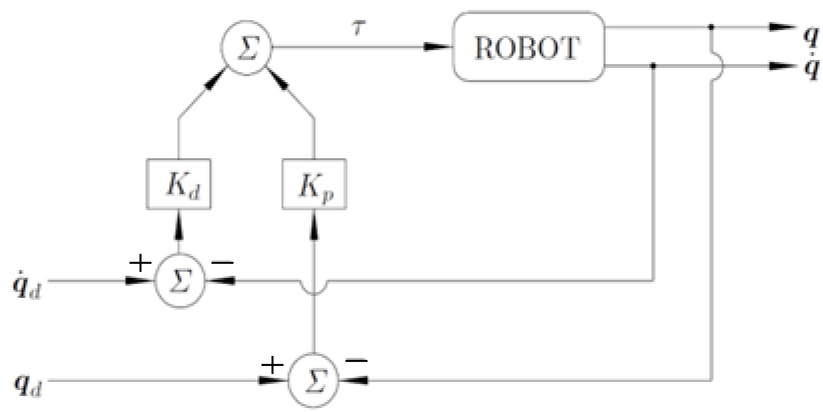

Figure 3.12: Diagram block of the derivative proportional PD controller of the manipulator.

a. Angles tracking

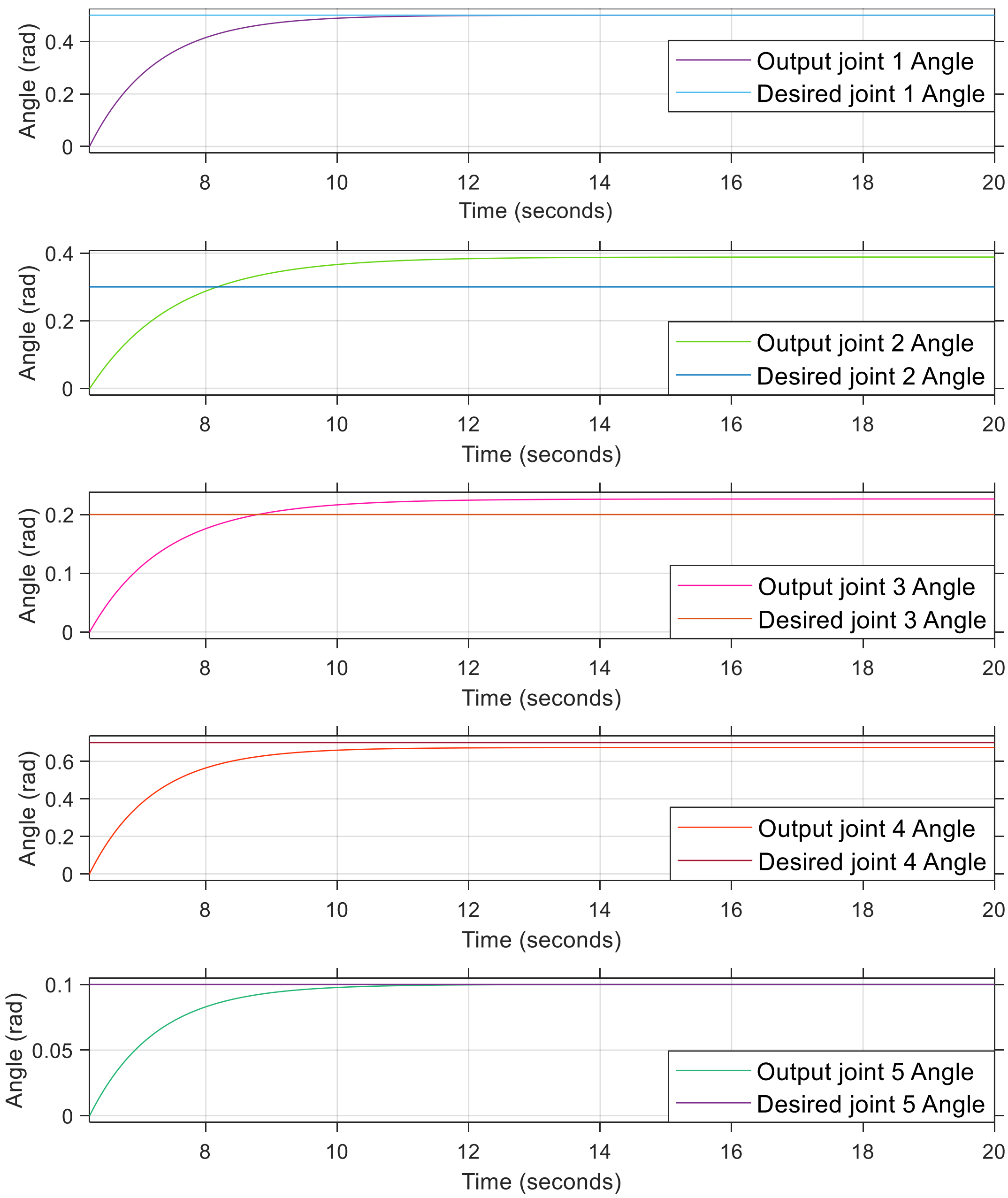

Figure 3.13: the desired and output joints angles result by the manipulator (PD controller case).

From this figure, we can notice that the response to the desired given joints angles by the manipulator is very bad and not accurate, that is what makes the PD controller not reliable when using small K_p gains.

b. Torques Commands

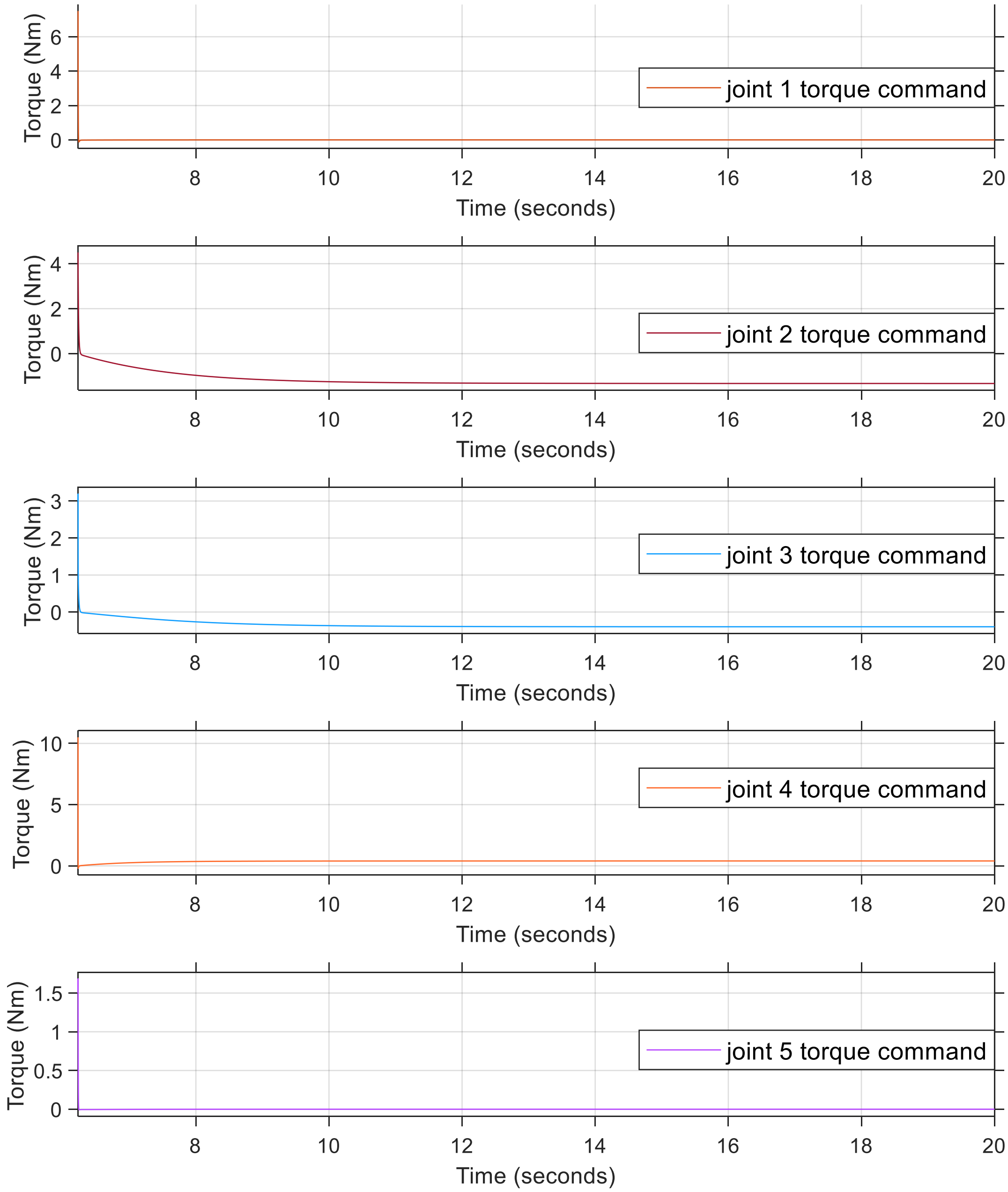

Figure 3.14: The Giving Torques Commands from the PD controller.

B. gravity compensation controller results

The control process:

This time we will use the PD controller with gravity compensation, also by using the feedback of both actual joints' angles (actual trajectory) $q(t)$ and the actual joints speeds $\dot{q}(t)$. The control law is given as follows: [11]

$$\tau = K_p e + K_d \dot{e} + g(q) \tag{3.24}$$

With $K_p \in \mathbb{R}^{5\times5}$, $K_d \in \mathbb{R}^{5\times5}$ are defined positive matrices [11], $\boldsymbol{q_d}$ is the trajectory reference (desired). After that, the final obtained joints torque $\tau = [\tau_1\ \tau_2\ \tau_3\ \tau_4\ \tau_5]^T$ will command and drive the manipulator joints to the desired trajectory. The structure of this control process is showed in the figure below: [11]

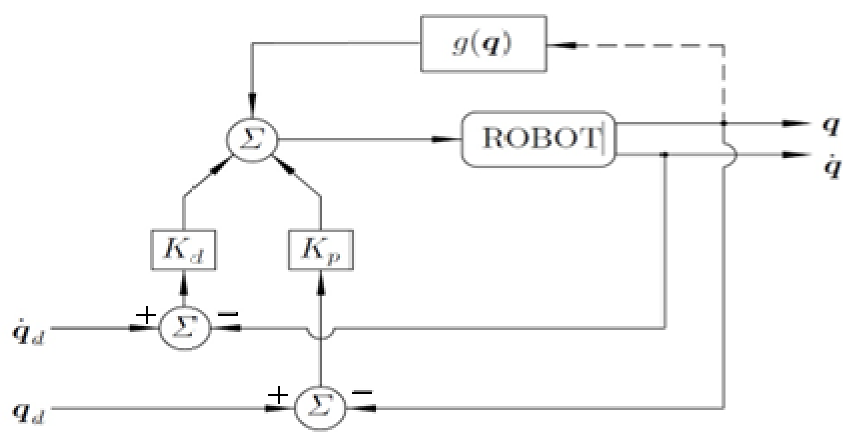

Figure 3.15: Diagram block of the PD controller with gravity compensation of the manipulator.

a. angles tracking

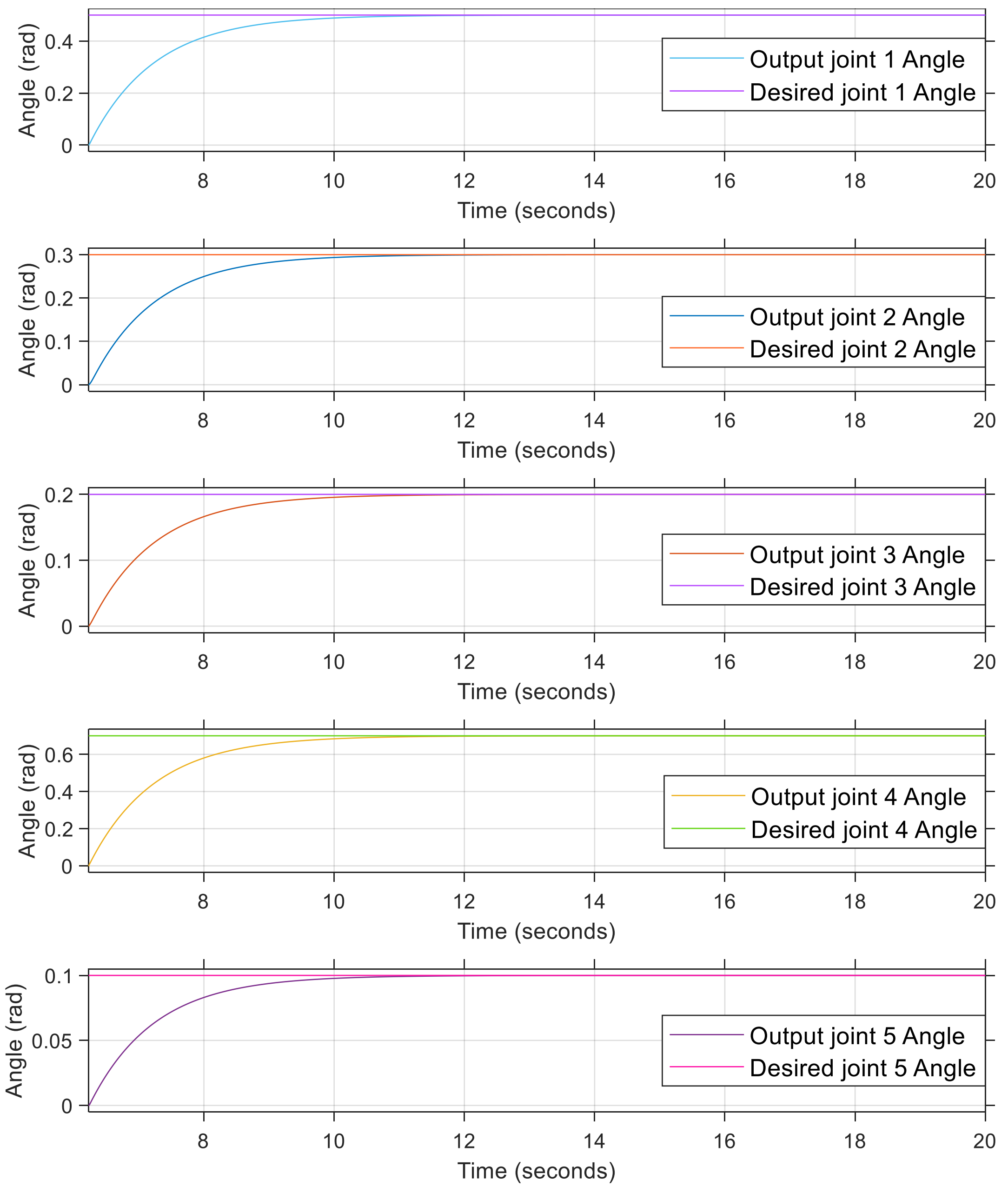

Figure 3.16: the desired and output joints angles result by the manipulator (gravity compensation controller case).

From this figure, we can conclude that the response to the desired given joints angles by the manipulator is very acceptable and accurate, which means that the gravity compensation controller is reliable, because it takes into account the gravity.

b. Torques Commands

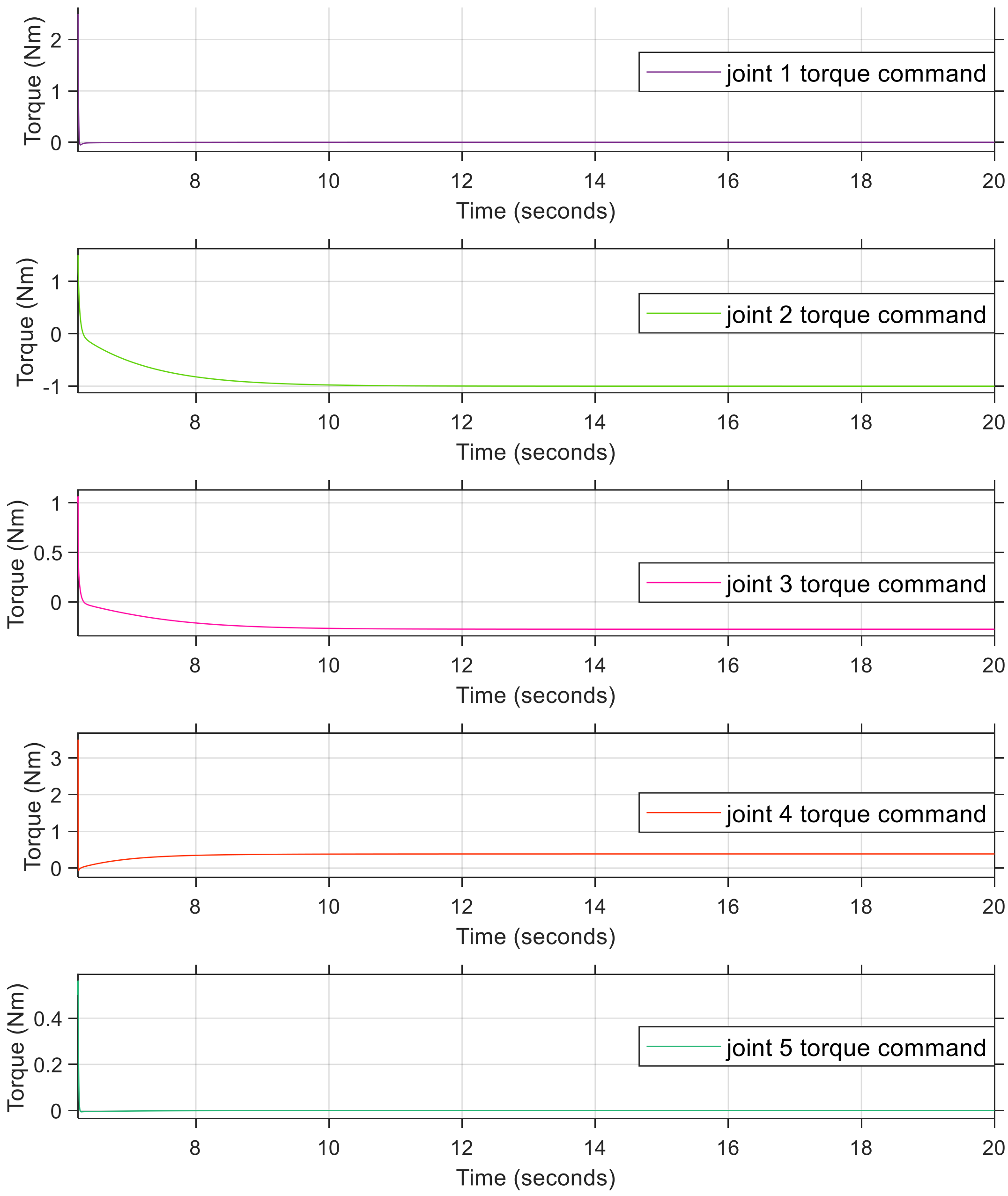

Figure 3.17: The Giving Torques Commands from the gravity compensation controller.

C. Computed-Torques controller results

The control process:

The lust controller will be the Computed-Torques controller, also by using the feedback of both actual joints' angles (actual trajectory) $q(t)$ and the actual joints speeds $\dot{q}(t)$. The control law is given as follows: [11]

$$\tau = M(q)\left(\ddot{q}_d + K_p e + K_d \dot{e}\right) + C(q, \dot{q})\dot{q} + G(q) \qquad (3.25)$$

With $K_p \in \mathbb{R}^{5\times5}$, $K_d \in \mathbb{R}^{5\times5}$ are defined positive matrices [11], $\boldsymbol{q_d}$ is the trajectory reference (desired). After that, the final obtained joints torque $\tau = [\tau_1\ \tau_2\ \tau_3\ \tau_4\ \tau_5]^T$ will command and drive the manipulator joints to the desired trajectory. The structure of this control process is showed in the figure below: [11]

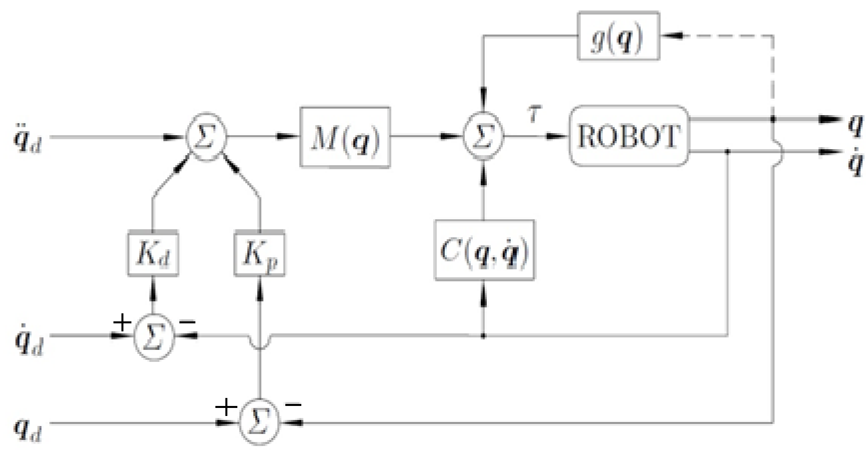

Figure 3.18: Diagram block of the Computed-Torques controller of the manipulator.

a. Angles tracking

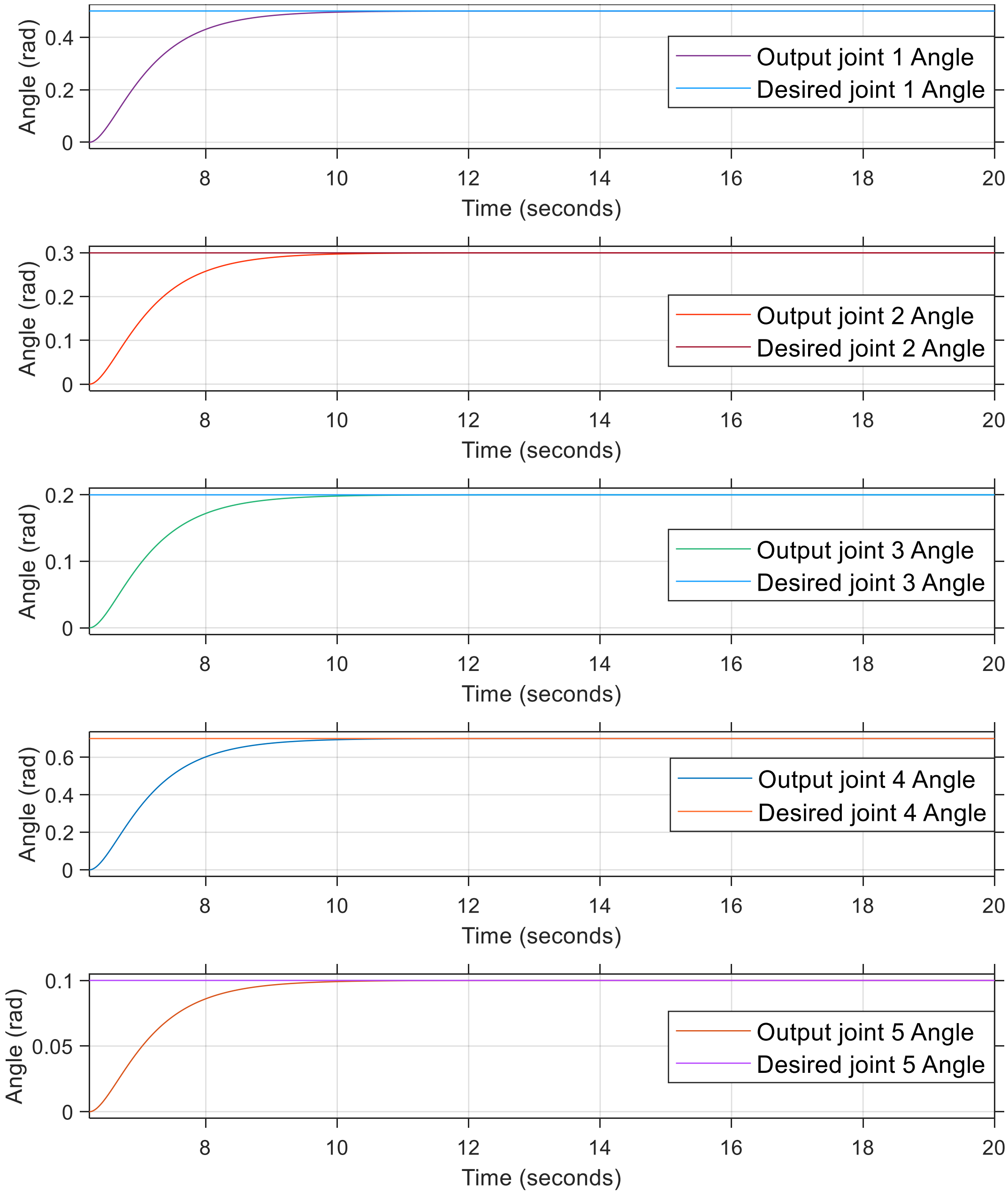

Figure 3.19: the desired and output joints angles result by the manipulator (Computed-Torques controller case).

After looking to this figure, we can clearly see that the response to the desired given joints angles by the manipulator is very smooth, acceptable and accurate, which means also that the Computed-Torques controller is effective.

b. Torques Commands

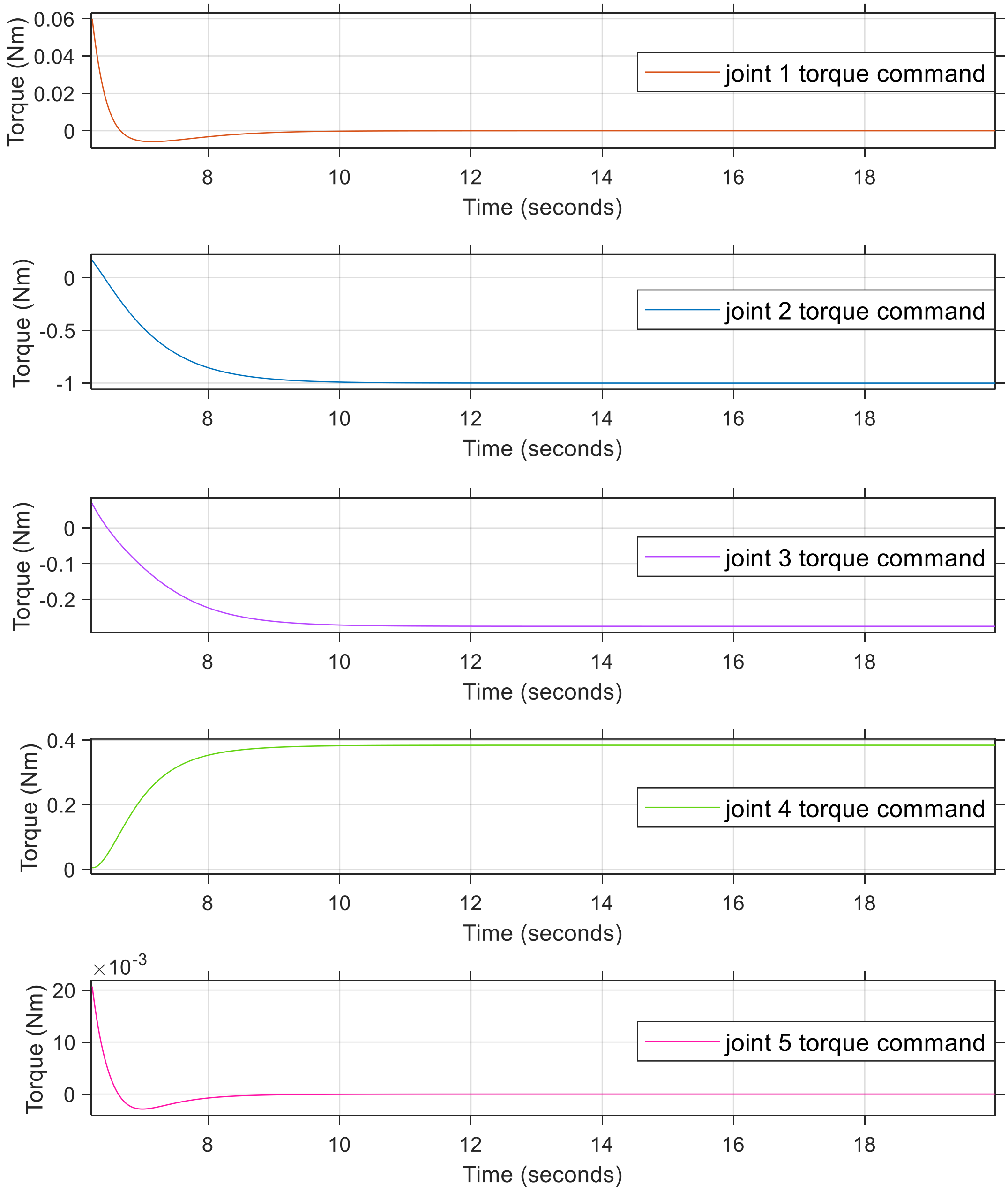

Figure 3.20: The Giving Torques Commands from the Computed-Torques controller.

2.3.2 Combined System

In this section, we will see the simulation results of controlling the combined system (assuming arm mounted on the platform) by a Fuzzy logic-PID controller to take advantage of its proprieties (non-linear systems controller and adaptative to perturbations). This time, we will give a desired trajectory that contains sequence of three positions and three orientations to our model and see how it will response to it.

2.3.2.1 The control process

In this time the control process is going to be in the operational space. To ensure the stability and accuracy of the whole control system, we have designed a fuzzy-PID controller for our robot. Figure (3.21) shows the block diagram of the overall system. We will use the Fuzzy logic-PID controller by using the feedback of both actual end-effector position and orientation. The input of fuzzy-PID controller is the error (e) and change in error (ec) [8]. The control law is given as follows: [1]

$$\dot{q}(t) = \dot{q}_d(t) + K_p(q_d(t) - q(t)) + K_i \int_0^t (q_d(\mathrm{t}) - q(\mathrm{t}))d\mathrm{t} \tag{3.26}$$

And

$$u = J^{-1}\dot{q} \tag{3.27}$$

Where: J^{-1} is the pseudo inverse of the Jacobian of the combined system given in equation (2.53).

The gains K_p and K_i have positive values and they are calculated continuously throughout the duration of the control, which makes it have the property of adapting to the perturbations. $q(t)$ is an estimate of the actual end-effector position and orientation derived from sensors. Then $\dot{q}(t)$ can be converted to the commanded wheel driving velocities and manipulator' joints velocities $u(t)$ using Equation (3.27). After that, the final obtained wheels speeds and joints velocities $u(t)$ will command and drive the end-effector to the desired waypoint. The structure of this control process is showed in the figure below:

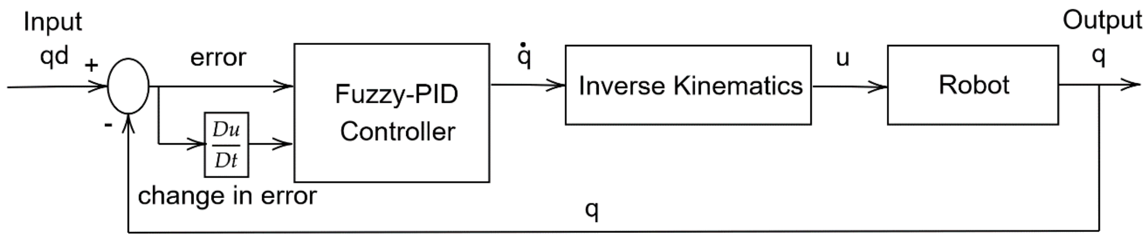

Figure 3.21: Diagram block of Fuzzy logic-PID controller and the combined system' kinematics (Simulink).

2.3.2.2 Design of controller with an explanation of the work steps

The figure (3.22) shows the components of the fuzzy logic controller which are: Fuzzification, Rule base (The inference engine) and Defuzzification. We are going to explain them step by step: [9]

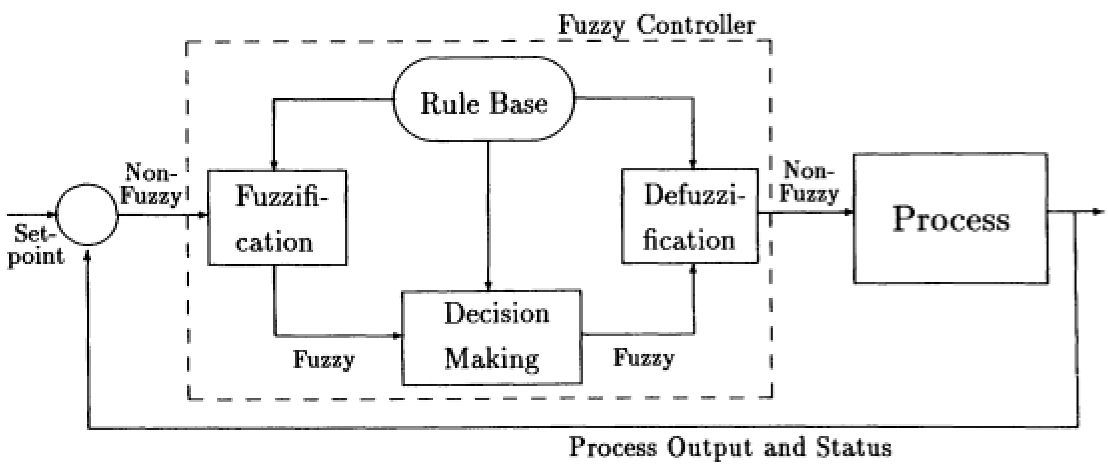

Figure 3.22: the components of the fuzzy logic controller.

a) Fuzzification:

The purpose of the fuzzification step is to transform a numerical data into a linguistic variable. For this, we must create membership functions. A membership function is a function which makes it possible to define the degree of membership of a numerical data item to a linguistic variable [10]. We have chosen two inputs (error and change in error) and three outputs (gains: K_p, K_i, K_d) Membership functions: [8]

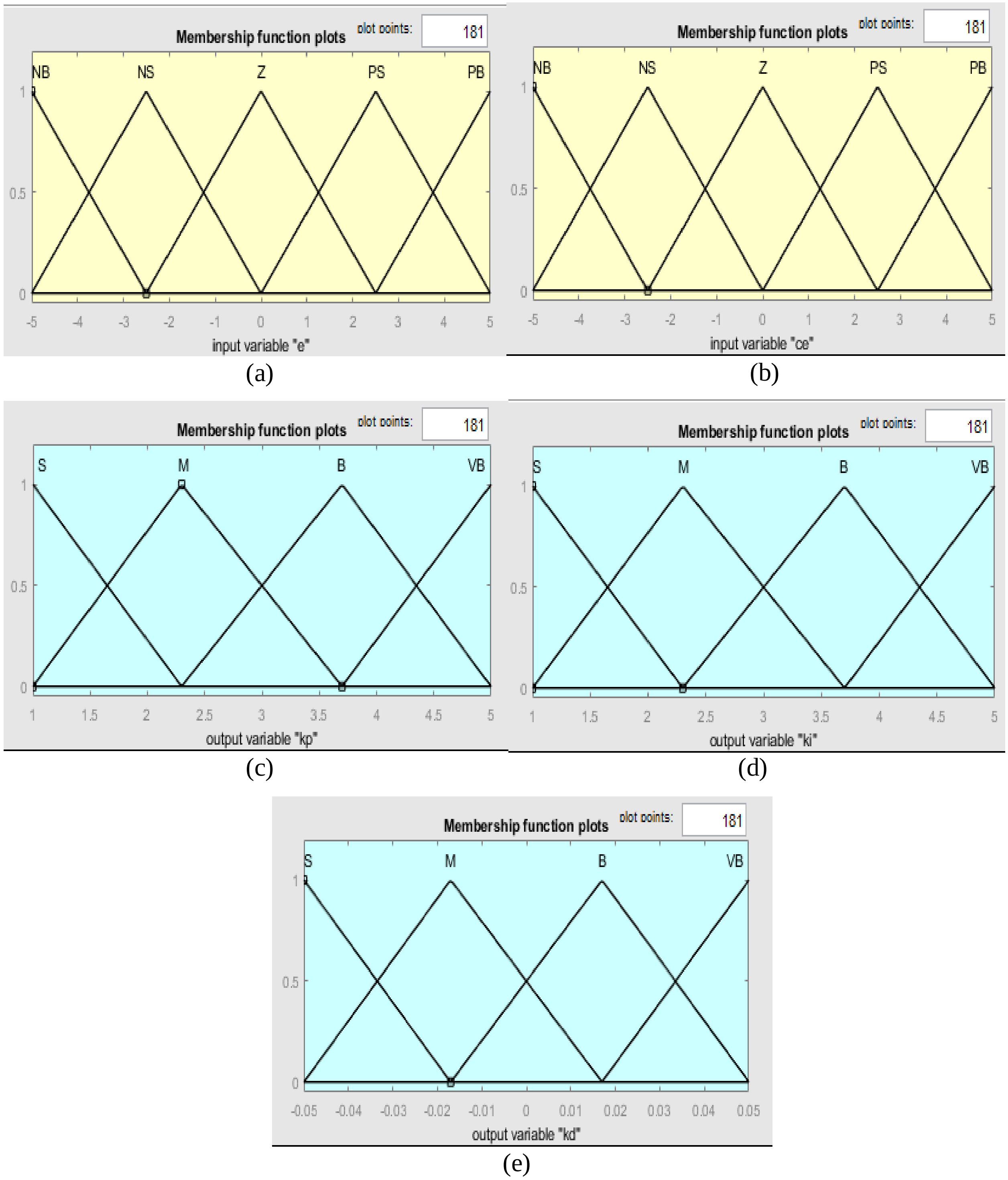

Figure 3.23: Membership functions of input and output. (a) 'e', (b) 'ec', (c) 'K_p', (d) 'K_i', (e) 'K_d'.

For choosing the variables of error (e) and change in error (ec), we set five fuzzy input values: negative big (NB), negative small (NS), zero (Z), positive small (PS), positive big (PB). Four output values are chosen: very big (VB), big (B), medium (M), small (S) (Table 3.1). The fuzzy-PID controller includes two inputs $(e(t), ec(t))$ and three outputs (K_p, K_i, K_d). It is applied to determine a parameter of PID controller with the membership function of a triangle form. The ranges of these membership function are determined by experience. The membership function of all output and input are described by Figure (3.23). [8]

b) Rule base (The inference engine):

Now that we have linguistic variables, we will be able to pass them into the inference engine. Here, each rule of the inference engine is written by the designer of the fuzzy system according to the knowledge he possesses [10], the used rule base showing in table (3.1) was taken from this paper [8]

e/ce	**NB**	**NS**	**Z**	**PS**	**PB**
NB	VB/S/S	VB/M/S	S/M/VB	S/M/B	M/S/S
NS	VB/B/S	B/B/S	S/B/B	S/B/B	B/B/S
Z	B/B/S	M/B/M	S/VB/B	S/VB/M	VB/VB/S
PS	M/B/M	S/B/B	S/B/B	M/B/S	VB/B/S
PB	S/S/M	S/S/VB	S/M/VB	B/M/S	VB/S/S

Table 3.1: Fuzzy rule table for $K_p/K_i/K_d$.

c) Defuzzification:

The last step to have an operational fuzzy system is called defuzzification. During the second step, we generated a bunch of commands in the form of linguistic variables (one command per rule). The purpose of defuzzification is to merge these commands and transform the resulting parameters into numerical data, for this we have used the method of the center of gravity. It consists in taking the abscissa corresponding to the center of gravity of the membership function. [10]

Where:

$$CG = \frac{\sum_{x=a}^{b} \mu_A(x) \cdot x}{\sum_{x=a}^{b} \mu_A(x)} \tag{3.27}$$

2.3.2.3 Simulation

The given desired positions and orientations to the end-effector are:

- position along the axe x: $[p_x(1) = 0\,; p_x(2) = 0.6\,;\ p_x(3) = 1]$.
- position along the axe y:$[p_y(1) = 0\,; p_y(2) = 1\,;\ p_y(3) = 1.2]$.
- position along the axe z: $[p_z(1) = 0\,; p_z(2) = 0.5\,;\ p_z(3) = 0.6]$.
- rotation angle around the axe z (Yaw): $[\alpha_1 = 0\,; \alpha_2 = 0.0.4\,;\ \alpha_3 = 0.12]$.
- rotation angle around the axe y (Pitch): $[\beta_1 = 0\,; \beta_2 = 0.02\,; \beta_3 = 0.13]$.
- rotation angle around the axe x (Roll): $[\gamma_1 = 0\,; \gamma_2 = 0\,;\ \gamma_3 = 0.14]$.

a. Position and Orientation of the end effector

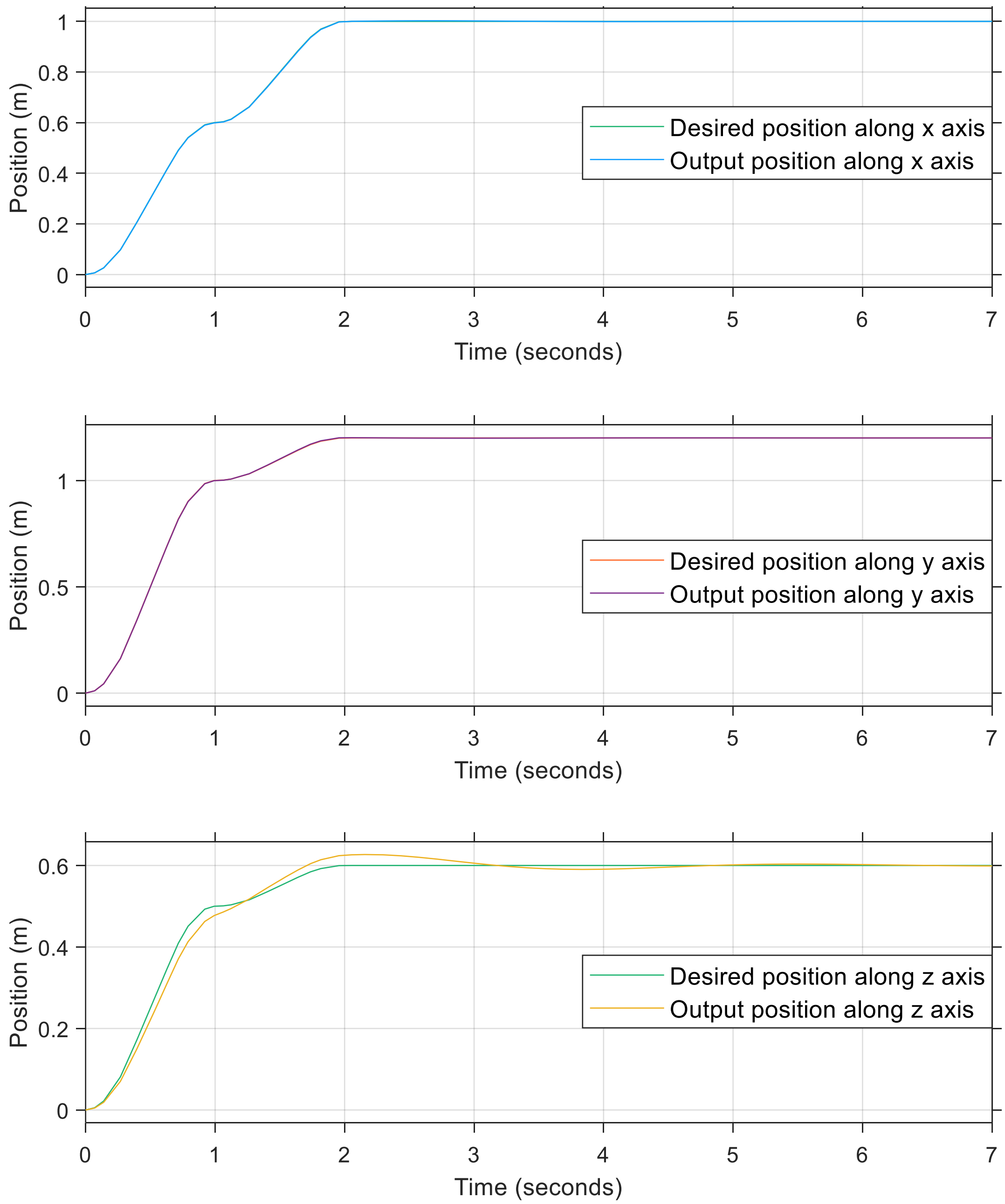

Figure 3.24: the desired and output Position of the end effector results.

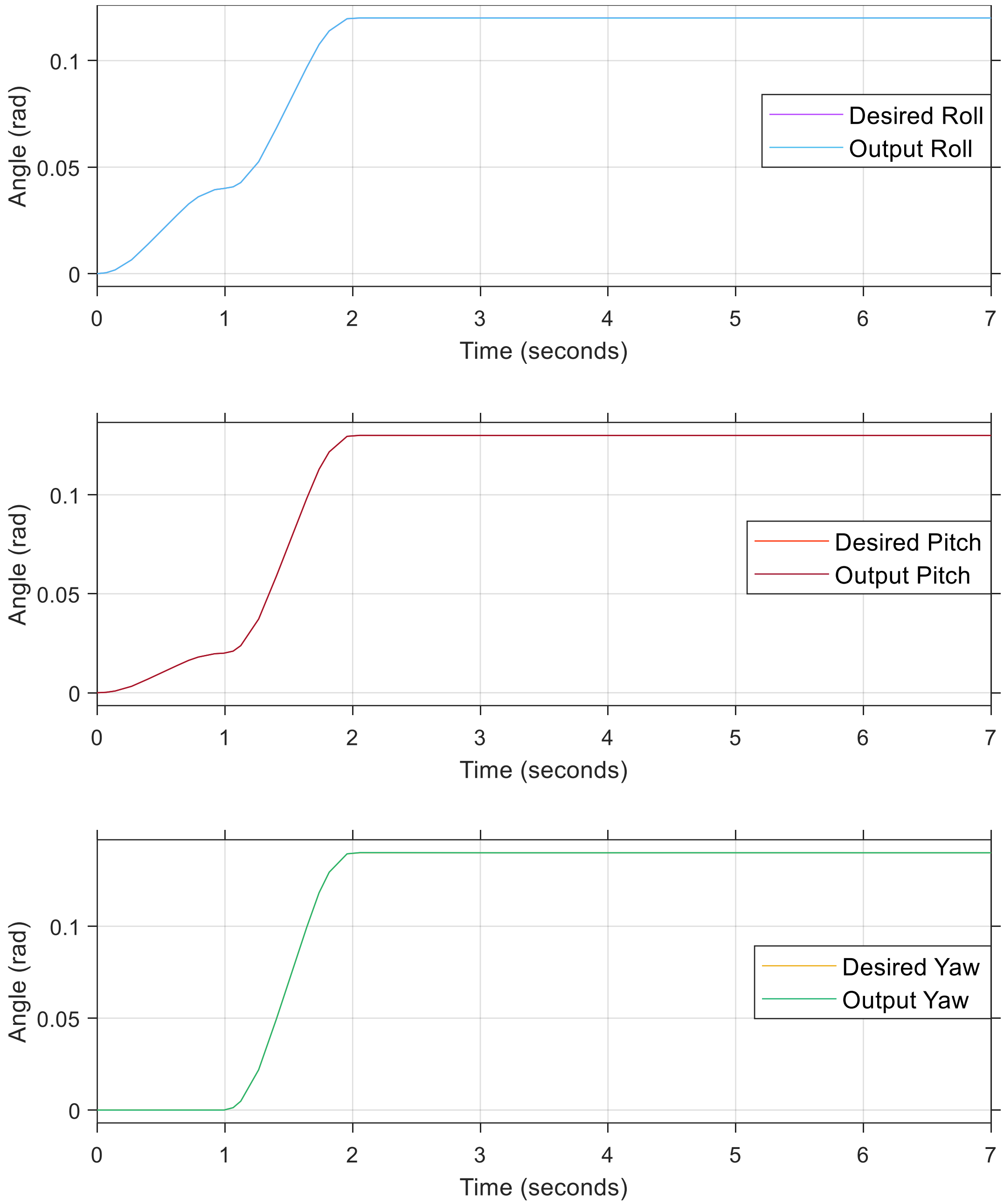

Figure 3.25: the desired and output Orientation of the end effector results.

From these figures, we can conclude that the response to the desired given positions and angles by the end-effector of the robot is very reasonable, smooth and accurate, which means that our combined system kinematic model is correct, and also the used Fuzzy logic-PID controller is effective and reliable.

b. Velocity commands

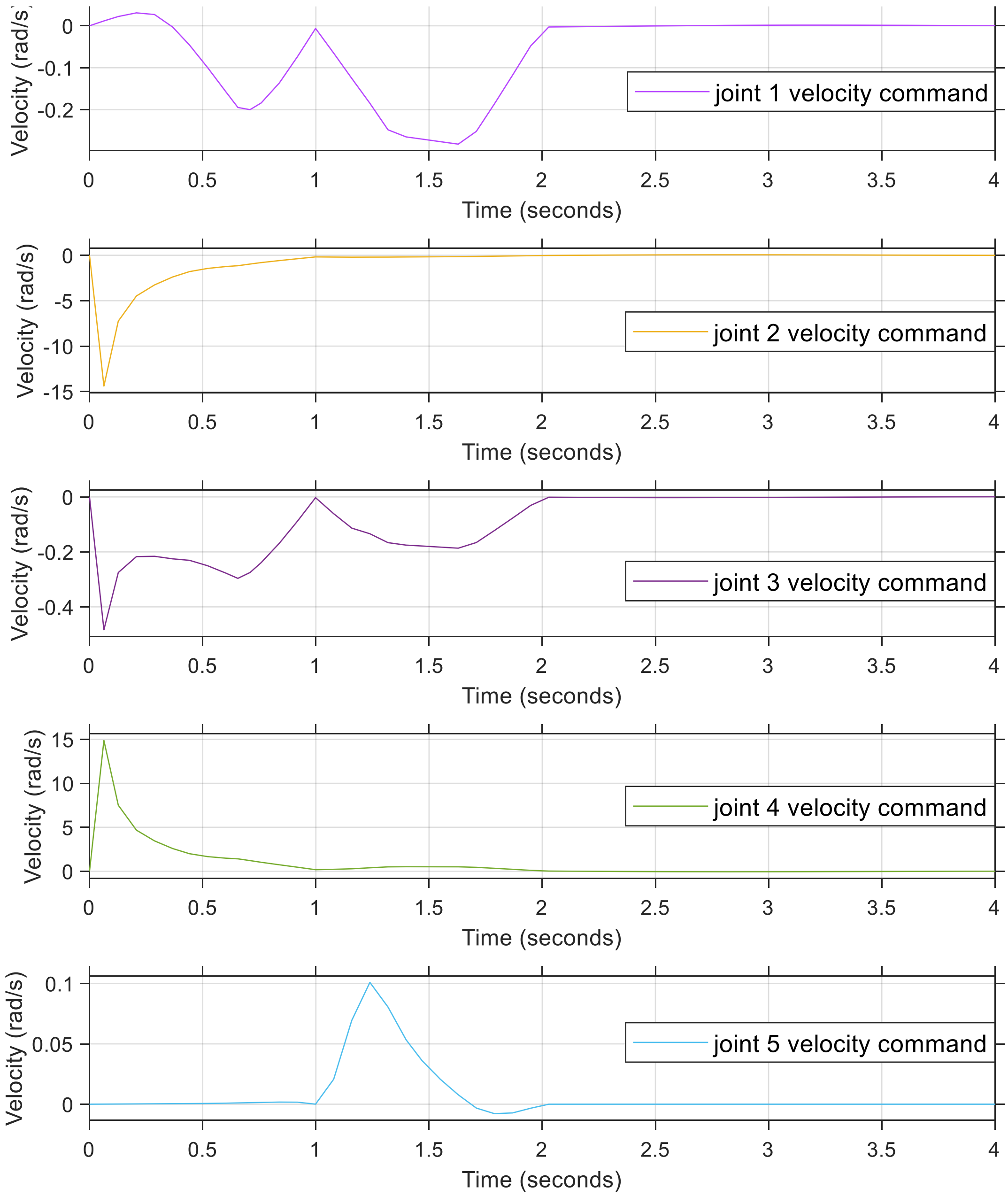

Figure 3.26: The Giving joints' velocity Commands from the controller.

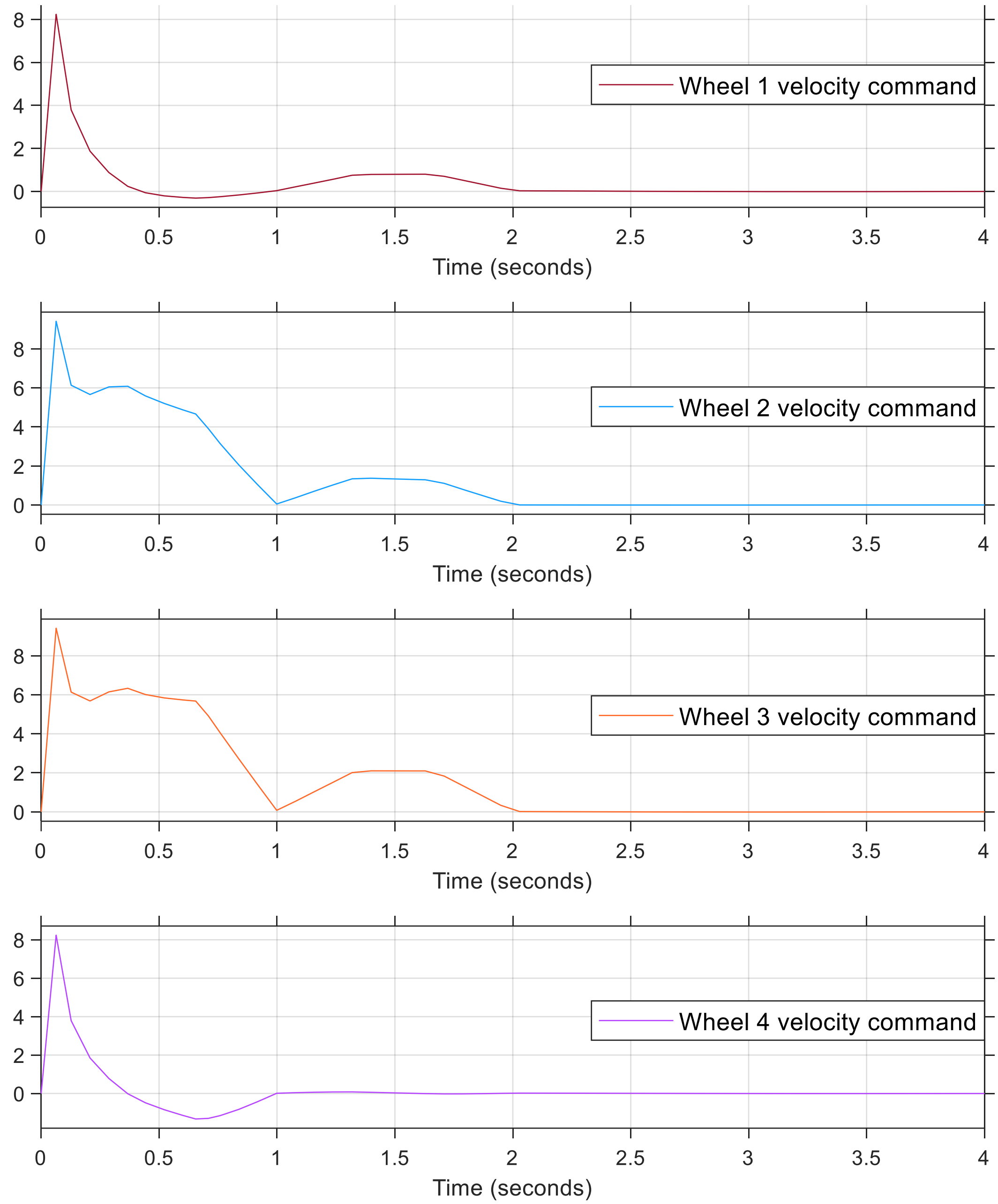

Figure 3.27: The Giving Wheels’ velocities Commands from the controller.

2.4 Conclusion

In this chapter, the analysis of the system response is achieved in the two parts. In the separated system, we gave a set of desired waypoints to the robot platform first, they were well tracked by the platform, after that the study of the manipulator was done in three cases using a new controller every time in order to see which one is the best for our control, starting with the PD controller that gave us a bad result, in the other hand the manipulator responded very well to the given desired joints angles in both remain controllers, the gravity compensation controller and the computed-torques controller. In the combined system, we also gave a desired sequence of positions and orientations to the robot end-effector, they were perfectly tracked by the end-effector due to the advantages of the Fuzzy logic-PID controller. In general, the results were acceptable.

General Conclusion

In this work, an introduction about the development of industry robots, Kuka youBot robot components, characteristics and uses were discussed. In addition, modeling our Kuka youbot robot was done in two parts, the first part was a separated system modeling that gave us two models: the mobile platform kinematic model and the manipulator forward kinematic and dynamic model (we used the dynamic model in the simulation). Then the second part is the combined system kinematic model (forward and deferential kinematics), we rather prefer the dynamic model in this case but unfortunately, it’s not available due to its complexity.

Moving on to the control part, where we have done the simulation in two main parts: the separated system and the combined system, in order to compare the deference between them. In the separated system simulation, the mobile platform starts to move to the desired location, after it parks the manipulator starts moving, which gave us a perfect result. Also, In the combined system simulation, the robot end-effector responses very well to the given positions and orientations. In general, we got a well response in both of them, meaning that our models are correct and our controllers are well tuned.

By going through details in the simulation part, we must mention some notes: we used the dynamic model for the manipulator because this model takes into account: the inertia, forces and gravity; Also, because this model gives a satisfactory response, especially when using the gravity compensation controller and computed-torques controller.

Therefore, future works can use these models to do a specific task using robotic simulation software like MATLAB and ROS (Robot Operating System).

References

[1] Kevin M. Lynch and Frank C. Park MODERN ROBOTICS MECHANICS, PLANNING, AND CONTROL, Cambridge University Press, May 2017.

[2] Siciliano, Sciavicco, Villani, and Oriolo. Robotics: Modeling, Planning, and Control. Advanced Textbooks in Control and Signal Processing. Springer 2009.

[3] Konstantinos I. Kyriakopoulos, KINEMATIC ANALYSIS OF A KUKA youBot. NATIONAL TECHNICAL UNIVERSITY OF ATHENS SCHOOL OF MECHANICAL ENGINEERS MECHANICAL CONSTRUCTION SECTOR AND AUTOMATIC CONTROL, July 2016.

[4] Bischoff, R., Huggenberger, U., & Prassler, E. KUKA youBot - a mobile manipulator for research and education. 2011.

[5] https://www.startertutorials.com/uml/importance-of-modeling.html.

[6] Ahcene HAMOUDI, Karim BERKANI, Conception et réalisation d'un robot mobile autonome. UMMTO Tizi Ouzou, faculté de génie électrique et d'informatique. 2016.

[7] https://behfaco.com/projects/manipulator-robot.

[8] Do Khac Tiep, Kinam Lee, Dae-Yeong Im, Bongwoo Kwak, and Young-Jae, Design of Fuzzy-PID Controller for Path Tracking of Mobile Robot with Differential Drive, Department of Electric and Control Engineering, Mokpo National University, Jeonnam, Korea, September 2018.

[9] Nasser Amaitik, the Basics of Fuzzy Systems Technology, Aston University, January 2020.

[10] http://www.ferdinandpiette.com/blog/2011/08/les-systemes-flous-le-fonctionnement.

[11] RAHMANI BELKACEM, Control of Manipulator Robots, Amar Telidji University of Laghouat. 2021.

Appendix

Appendix (1):

dynamic model results:

- **the elements of the matrix $\boldsymbol{M}(\boldsymbol{q})$:**

$$M_{11} = 0.00340593 * sin(q2 + q3 + q4) - 0.000041499 * cos(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) - 0.00733 * cos(2.0 * q2 + 0.30948) - 0.000041 * cos(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) - 0.01418 * cos(2.0 * q2 + q3) + 0.00696599 * cos(2.0 * q2 + 2.0 * q3 + q4) - 0.02724 * cos(2.0 * q2) + 0.00008299 * cos(2.0 * q5) - 0.03305 * cos(q2 - 1.5313099) - 0.02083399 * cos(2.0 * q2 + q3 + 0.68828) + 0.0330444 * cos(q3 + 0.41206) - 0.007998699 * cos(q3 + q4) + 0.007998699 * cos(2.0 * q2 + q3 + q4) - 0.006966599 * cos(q4) - 0.00809 * cos(2.0 * q2 + 2.0 * q3) - 0.014070499 * cos(q2 + q3 - 1.158727) - 0.00146 * cos(2.0 * q2 + 2.0 * q3 + 2.0 * q4) + 0.07709799$$

$$M_{12} = 0.00008299 * cos(q2 + q3 + q4 + 2.0 * q5) - 0.00008 * cos(q2 + q3 + q4 - 2.0 * q5) - 0.00097 * cos(q2 + q3 + q4) + 0.00316767 * cos(q2 + 0.11886) + 0.00302 * cos(q2 + q3 - 0.6)$$

$$M_{13} = 0.00008299 * cos(q2 + q3 + q4 + 2.0 * q5) - 0.00008299 * cos(q2 + q3 + q4 - 2.0 * q5) - 0.00097799 * cos(q2 + q3 + q4) + 0.00302 * cos(q2 + q3 - 0.6008)$$

$$M_{14} = 0.00008299 * cos(q2 + q3 + q4 + 2.0 * q5) - 0.00008299 * cos(q2 + q3 + q4 - 2.0 * q5) - 0.00097799 * cos(q2 + q3 + q4)$$

$$M_{15} = 0.00689 * cos(q2 + q3 + q4)$$

$$M_{21} = 0.00008299 * cos(q2 + q3 + q4 + 2.0 * q5) - 0.00008 * cos(q2 + q3 + q4 - 2.0 * q5) - 0.00097 * cos(q2 + q3 + q4) + 0.00316767 * cos(q2 + 0.11886) + 0.00302 * cos(q2 + q3 - 0.6)$$

$$M_{22} = 0.06055 * cos(q3) - 0.01393 * cos(q4) - 0.02646 * sin(q3) - 0.0159 * cos(q3) * cos(q4) + 0.01599 * sin(q3) * sin(q4) - 0.00033199 * cos(q5)^2 + 0.1168039899$$

$$M_{23} = 0.0302784 * cos(q3) - 0.01393299 * cos(q4) - 0.01323399 * sin(q3) - 0.00799877499 * cos(q3) * cos(q4) + 0.00799877499 * sin(q3) * sin(q4) - 0.00033199 * cos(q5)^2 + 0.042026099$$

$$M_{24} = 0.00799799 * sin(q3) * sin(q4) - 0.00799799 * cos(q3) * cos(q4) - 0.006966599 * cos(q4) - 0.00033199 * cos(q5)^\wedge 2 + 0.004928$$

$$M_{25} = 0$$

$$M_{31} = 0.0000829999 * cos(q2 + q3 + q4 + 2.0 * q5) - 0.00008299999 * cos(q2 + q3 + q4 - 2.0 * q5) - 0.000977999 * cos(q2 + q3 + q4) + 0.00302 * cos(q2 + q3 - 0.6008)$$

$$M_{32} = 0.03027840 * cos(q3) - 0.013932999 * cos(q4) - 0.013233999 * sin(q3) - 0.007998774 * cos(q3) * cos(q4) + 0.007998774 * sin(q3) * sin(q4) - 0.00033199 * cos(q5)^\wedge 2 + 0.0420260$$

$$M_{33} = 0.042026189 - 0.000331999 * cos(q5)^\wedge 2 - 0.01393335 * cos(q4)$$

$$M_{34} = 0.00492868 - 0.00033199 * cos(q5)^\wedge 2 - 0.006966675 * cos(q4)$$

$$M_{35} = 0$$

$$M_{41} = 0.0000829999999 * cos(q2 + q3 + q4 + 2.0 * q5) - 0.0000829999 * cos(q2 + q3 + q4 - 2.0 * q5) - 0.00097799 * cos(q2 + q3 + q4)$$

$$M_{42} = 0.0079979999 * sin(q3) * sin(q4) - 0.007997999 * cos(q3) * cos(q4) - 0.006966599 * cos(q4) - 0.000331999 * cos(q5)^\wedge 2 + 0.004928$$

$$M_{43} = 0.00492868 - 0.0003319999 * cos(q5)^\wedge 2 - 0.0069666750 * cos(q4)$$

$$M_{44} = 0.0049286 - 0.0003319999 * cos(q5)^\wedge 2$$

$$M_{45} = 0$$

$$M_{51} = 0.00689 * cos(q2 + q3 + q4)$$

$$M_{52} = 0$$

$$M_{53} = 0$$

$$M_{54} = 0$$

$$M_{55} = 0.00689$$

- **the elements of the matrix $C(q, \dot{q})$:**

$$
\begin{aligned}
C_{11} = {} & 0.01418 * dq2 * sin(2.0 * q2 + q3) + 0.00709 * dq3 * sin(2.0 * q2 + q3) \\
& - 0.00696599 * dq2 * sin(2.0 * q2 + 2.0 * q3 + q4) - 0.006965999 \\
& * dq3 * sin(2.0 * q2 + 2.0 * q3 + q4) - 0.00348299 * dq4 * sin(2.0 * q2 \\
& + 2.0 * q3 + q4) + 0.02724 * dq2 * sin(2.0 * q2) - 0.0000829999 * dq5 \\
& * sin(2.0 * q5) + 0.016525 * dq2 * sin(q2 - 1.531309) + 0.020833999 \\
& * dq2 * sin(2.0 * q2 + q3 + 0.68828) + 0.0104169999 * dq3 * sin(2.0 \\
& * q2 + q3 + 0.68828) - 0.0165222000000 * dq3 * sin(q3 + 0.41206) \\
& + 0.00399934 * dq3 * sin(q3 + q4) + 0.0039993499 * dq4 * sin(q3 \\
& + q4) - 0.007998699999 * dq2 * sin(2.0 * q2 + q3 + q4) \\
& - 0.003999349999 * dq3 * sin(2.0 * q2 + q3 + q4) - 0.00399934 * dq4 \\
& * sin(2.0 * q2 + q3 + q4) + 0.0034832999 * dq4 * sin(q4) + 0.00809 \\
& * dq2 * sin(2.0 * q2 + 2.0 * q3) + 0.00809 * dq3 * sin(2.0 * q2 + 2.0 \\
& * q3) + 0.0070352499 * dq2 * sin(q2 + q3 - 1.158727) \\
& + 0.0070352499 * dq3 * sin(q2 + q3 - 1.158727000) + 0.00146 * dq2 \\
& * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4) + 0.00146 * dq3 * sin(2.0 * q2 + 2.0 \\
& * q3 + 2.0 * q4) + 0.00146 * dq4 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4) \\
& + 0.00004100 * dq2 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) \\
& + 0.0000414999 * dq2 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) \\
& + 0.00004100000 * dq3 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) \\
& + 0.00004149999 * dq3 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) \\
& + 0.000041000 * dq4 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) \\
& + 0.000041499999 * dq4 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) \\
& - 0.00004100 * dq5 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) \\
& + 0.000041499999 * dq5 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) \\
& + 0.001702965 * dq2 * cos(q2 + q3 + q4) + 0.001702965 * dq3 \\
& * cos(q2 + q3 + q4) + 0.001702965 * dq4 * cos(q2 + q3 + q4) \\
& + 0.00733 * dq2 * sin(2.0 * q2 + 0.30948)
\end{aligned}
$$

$$C_{12} = 0.01418 * dq1 * sin(2.0 * q2 + q3) - 0.0069659999 * dq1 * sin(2.0 * q2 + 2.0 * q3 + q4) + 0.02724 * dq1 * sin(2.0 * q2) + 0.016525 * dq1 * sin(q2 - 1.531309999) + 0.020833 * dq1 * sin(2.0 * q2 + q3 + 0.68828) - 0.00316767 * dq2 * sin(q2 + 0.11886) - 0.0079986 * dq1 * sin(2.0 * q2 + q3 + q4) + 0.00809 * dq1 * sin(2.0 * q2 + 2.0 * q3) + 0.00703524 * dq1 * sin(q2 + q3 - 1.158727) - 0.00302 * dq2 * sin(q2 + q3 - 0.6) - 0.00151 * dq3 * sin(q2 + q3 - 0.6) - 0.00151 * dq3 * sin(q2 + q3 - 0.6008) + 0.00146 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4) + 0.0000410 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) + 0.00004149 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) + 0.00170296 * dq1 * cos(q2 + q3 + q4) + 0.00097 * dq2 * sin(q2 + q3 + q4) + 0.00097399999 * dq3 * sin(q2 + q3 + q4) + 0.000973999 * dq4 * sin(q2 + q3 + q4) - 0.003445 * dq5 * sin(q2 + q3 + q4) + 0.00733 * dq1 * sin(2.0 * q2 + 0.30948) + 0.00008 * dq2 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00008299 * dq2 * sin(q2 + q3 + q4 + 2.0 * q5) + 0.0000814999 * dq3 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00008299999 * dq3 * sin(q2 + q3 + q4 + 2.0 * q5) + 0.00008149999 * dq4 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00008299 * dq4 * sin(q2 + q3 + q4 + 2.0 * q5) - 0.00008 * dq5 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00008299 * dq5 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{13} = 0.00709 * dq1 * sin(2.0 * q2 + q3) - 0.00696599 * dq1 * sin(2.0 * q2 + 2.0 * q3 + q4) + 0.01041699 * dq1 * sin(2.0 * q2 + q3 + 0.68828) - 0.0165222 * dq1 * sin(q3 + 0.41206) + 0.00399934 * dq1 * sin(q3 + q4) - 0.00399934 * dq1 * sin(2.0 * q2 + q3 + q4) + 0.00809 * dq1 * sin(2.0 * q2 + 2.0 * q3) + 0.00703524 * dq1 * sin(q2 + q3 - 1.158727) - 0.00151 * dq2 * sin(q2 + q3 - 0.6) - 0.00151 * dq2 * sin(q2 + q3 - 0.6008) - 0.00302 * dq3 * sin(q2 + q3 - 0.6008) + 0.00146 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4) + 0.000041 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) + 0.00004149 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) + 0.001702965 * dq1 * cos(q2 + q3 + q4) + 0.0009739999 * dq2 * sin(q2 + q3 + q4) + 0.00097799 * dq3 * sin(q2 + q3 + q4) + 0.00097799 * dq4 * sin(q2 + q3 + q4) - 0.003445 * dq5 * sin(q2 + q3 + q4) + 0.00008149999 * dq2 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.000082999999 * dq2 * sin(q2 + q3 + q4 + 2.0 * q5) + 0.000082999 * dq3 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.0000829999 * dq3 * sin(q2 + q3 + q4 + 2.0 * q5) + 0.0000829999 * dq4 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.0000829 * dq4 * sin(q2 + q3 + q4 + 2.0 * q5) - 0.0000829999 * dq5 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.0000829999 * dq5 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{14} = 0.00399934 * dq1 * sin(q3 + q4) - 0.003482999 * dq1 * sin(2.0 * q2 + 2.0 * q3 + q4) - 0.0039993 * dq1 * sin(2.0 * q2 + q3 + q4) + 0.003483299 * dq1 * sin(q4) + 0.00146 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4) + 0.00004100 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) + 0.0000414999 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) + 0.0017029650 * dq1 * cos(q2 + q3 + q4) + 0.0009739999 * dq2 * sin(q2 + q3 + q4) + 0.00097799 * dq3 * sin(q2 + q3 + q4) + 0.00097799 * dq4 * sin(q2 + q3 + q4) - 0.003445 * dq5 * sin(q2 + q3 + q4) + 0.000081499999 * dq2 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.0000829999 * dq2 * sin(q2 + q3 + q4 + 2.0 * q5) + 0.0000829999999 * dq3 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00008299999 * dq3 * sin(q2 + q3 + q4 + 2.0 * q5) + 0.0000829999 * dq4 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00008299 * dq4 * sin(q2 + q3 + q4 + 2.0 * q5) - 0.00008299999 * dq5 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00008299999 * dq5 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{15} = -0.5 * dq2 * (0.00689 * sin(q2 + q3 + q4) + 0.00016 * sin(q2 + q3 + q4 - 2.0 * q5) + 0.00016599999 * sin(q2 + q3 + q4 + 2.0 * q5)) - 0.0000000000000000000677626 * dq1 * (6050532056176733.0 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) - 6124319032471571.0 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) + 12248638064943142.0 * sin(2.0 * q5)) - 0.5 * dq3 * (0.00689 * sin(q2 + q3 + q4) + 0.000165999 * sin(q2 + q3 + q4 - 2.0 * q5) + 0.0001659999 * sin(q2 + q3 + q4 + 2.0 * q5)) - 0.5 * dq4 * (0.00689 * sin(q2 + q3 + q4) + 0.00016599 * sin(q2 + q3 + q4 - 2.0 * q5) + 0.00016599999 * sin(q2 + q3 + q4 + 2.0 * q5))$$

$$C_{21} = 0.00696599 * dq1 * sin(2.0 * q2 + 2.0 * q3 + q4) - 0.01418 * dq1 * sin(2.0 * q2 + q3) - 0.02724 * dq1 * sin(2.0 * q2) - 0.016525 * dq1 * sin(q2 - 1.531309) - 0.0208339 * dq1 * sin(2.0 * q2 + q3 + 0.68828) + 0.0079986 * dq1 * sin(2.0 * q2 + q3 + q4) - 0.00809 * dq1 * sin(2.0 * q2 + 2.0 * q3) - 0.007035249 * dq1 * sin(q2 + q3 - 1.15872700) - 0.00151 * dq3 * sin(q2 + q3 - 0.6) + 0.00151 * dq3 * sin(q2 + q3 - 0.6008) - 0.00146 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4) - 0.0000410000 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) - 0.00004149 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) - 0.001702965 * dq1 * cos(q2 + q3 + q4) - 0.0000039999 * dq3 * sin(q2 + q3 + q4) - 0.000003999 * dq4 * sin(q2 + q3 + q4) + 0.003445 * dq5 * sin(q2 + q3 + q4) - 0.00733 * dq1 * sin(2.0 * q2 + 0.30948) - 0.0000014999 * dq3 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.000001499 * dq4 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00008 * dq5 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00008299 * dq5 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{22} = 0.5 * dq4 * (0.000045 * sin(q3 - 1.0 * q4) + 0.015945 * sin(q3 + q4) + 0.01393 * sin(q4)) + 0.0001659999 * dq5 * sin(2.0 * q5) - 0.5 * dq3 * (0.000045 * sin(q3 - 1.0 * q4) - 0.015945 * sin(q3 + q4) + 0.02646 * cos(q3) + 0.06055 * sin(q3))$$

$$C_{23} = 0.0001659999 * dq5 * sin(2.0 * q5) - 0.0000225 * dq2 * sin(q3 - 1.0 * q4) + 0.0079725 * dq2 * sin(q3 + q4) + 0.00799877 * dq3 * sin(q3 + q4) + 0.0079983 * dq4 * sin(q3 + q4) - 0.01323 * dq2 * cos(q3) - 0.013233 * dq3 * cos(q3) - 0.030275 * dq2 * sin(q3) - 0.0302784 * dq3 * sin(q3) + 0.0069664 * dq4 * sin(q4) - 0.00151 * dq1 * sin(q2 + q3 - 0.6) + 0.00151 * dq1 * sin(q2 + q3 - 0.6008) - 0.000003999 * dq1 * sin(q2 + q3 + q4) - 0.00000149999 * dq1 * sin(q2 + q3 + q4 - 2.0 * q5)$$

$$C_{24} = 0.0000225 * dq2 * sin(q3 - 1.0 * q4) + 0.00016599 * dq5 * sin(2.0 * q5) + 0.0079725 * dq2 * sin(q3 + q4) + 0.007998387 * dq3 * sin(q3 + q4) + 0.00799799 * dq4 * sin(q3 + q4) + 0.006965 * dq2 * sin(q4) + 0.00696649 * dq3 * sin(q4) + 0.0069665 * dq4 * sin(q4) - 0.00000399999 * dq1 * sin(q2 + q3 + q4) - 0.00000149999 * dq1 * sin(q2 + q3 + q4 - 2.0 * q5)$$

$$C_{25} = 0.00016599999 * dq2 * sin(2.0 * q5) + 0.000165999 * dq3 * sin(2.0 * q5) + 0.000165999 * dq4 * sin(2.0 * q5) + 0.003445 * dq1 * sin(q2 + q3 + q4) - 0.00008 * dq1 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.0000829999 * dq1 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{31} = 0.00696599 * dq1 * sin(2.0 * q2 + 2.0 * q3 + q4) - 0.00709 * dq1 * sin(2.0 * q2 + q3) - 0.010416999 * dq1 * sin(2.0 * q2 + q3 + 0.68828) + 0.0165222 * dq1 * sin(q3 + 0.41206) - 0.00399934 * dq1 * sin(q3 + q4) + 0.00399934 * dq1 * sin(2.0 * q2 + q3 + q4) - 0.00809 * dq1 * sin(2.0 * q2 + 2.0 * q3) - 0.00703524 * dq1 * sin(q2 + q3 - 1.1587270) + 0.00151 * dq2 * sin(q2 + q3 - 0.6) - 0.00151 * dq2 * sin(q2 + q3 - 0.6008) - 0.00146 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4) - 0.00004100 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) - 0.000041499 * dq1 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) - 0.00170296 * dq1 * cos(q2 + q3 + q4) + 0.0000039999 * dq2 * sin(q2 + q3 + q4) + 0.003445 * dq5 * sin(q2 + q3 + q4) + 0.000001499999 * dq2 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00008299 * dq5 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.000082999 * dq5 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{32} = 0.0000225 * dq2 * sin(q3 - 1.0 * q4) + 0.00016599 * dq5 * sin(2.0 * q5) - 0.0079725 * dq2 * sin(q3 + q4) + 0.00000038749 * dq4 * sin(q3 + q4) + 0.01323 * dq2 * cos(q3) + 0.030275 * dq2 * sin(q3) + 0.00696649 * dq4 * sin(q4) + 0.00151 * dq1 * sin(q2 + q3 - 0.6) - 0.00151 * dq1 * sin(q2 + q3 - 0.6008) + 0.0000039999 * dq1 * sin(q2 + q3 + q4) + 0.0000014999 * dq1 * sin(q2 + q3 + q4 - 2.0 * q5)$$

$$C_{33} = 0.000165999 * dq5 * sin(2.0 * q5) + 0.0069666 * dq4 * sin(q4)$$

$$C_{34} = 0.0001659999 * dq5 * sin(2.0 * q5) + 0.5 * dq2 * (0.000000774 * sin(q3 + q4) + 0.0139329 * sin(q4)) + 0.0069666 * dq3 * sin(q4) + 0.006966 * dq4 * sin(q4)$$

$$C_{35} = 0.0001659999 * dq2 * sin(2.0 * q5) + 0.0001659999 * dq3 * sin(2.0 * q5) + 0.000165999 * dq4 * sin(2.0 * q5) + 0.003445 * dq1 * sin(q2 + q3 + q4) - 0.0000829999 * dq1 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.0000829999 * dq1 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{41} = 0.5 * dq2 * (0.00000799 * sin(q2 + q3 + q4) + 0.000002999 * sin(q2 + q3 + q4 - 2.0 * q5)) - 0.5 * dq1 * (0.00292 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4) + 0.000082000 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) + 0.000082999 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) + 0.0034059 * cos(q2 + q3 + q4) - 0.006965 * sin(2.0 * q2 + 2.0 * q3 + q4) + 0.0079986 * sin(q3 + q4) - 0.00799869 * sin(2.0 * q2 + q3 + q4) + 0.006966599 * sin(q4)) - 0.5 * dq5 * (0.0001659999 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00689 * sin(q2 + q3 + q4) + 0.000165999 * sin(q2 + q3 + q4 + 2.0 * q5))$$

$$C_{42} = 0.000165999 * dq5 * sin(2.0 * q5) - 0.0000225 * dq2 * sin(q3 - 1.0 * q4) - 0.0079725 * dq2 * sin(q3 + q4) - 0.00000038749 * dq3 * sin(q3 + q4) - 0.006965 * dq2 * sin(q4) - 0.006966499999 * dq3 * sin(q4) + 0.00000399999 * dq1 * sin(q2 + q3 + q4) + 0.00000149999 * dq1 * sin(q2 + q3 + q4 - 2.0 * q5)$$

$$C_{43} = 0.0001659999 * dq5 * sin(2.0 * q5) - 0.5 * dq2 * (0.0000007749 * sin(q3 + q4) + 0.01393299 * sin(q4)) - 0.0069666 * dq3 * sin(q4)$$

$$C_{44} = 0.0001659999 * dq5 * sin(2.0 * q5)$$

$$C_{45} = 0.000165999 * dq2 * sin(2.0 * q5) + 0.00016599999 * dq3 * sin(2.0 * q5) + 0.0001659999 * dq4 * sin(2.0 * q5) + 0.003445 * dq1 * sin(q2 + q3 + q4) - 0.0000829999 * dq1 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.0000829999 * dq1 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{51} = 0.5 * dq2 * (0.00016 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00689 * sin(q2 + q3 + q4) + 0.0001659999 * sin(q2 + q3 + q4 + 2.0 * q5)) + 0.5 * dq1 * (0.000082000 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 - 2.0 * q5) - 0.0000829999999 * sin(2.0 * q2 + 2.0 * q3 + 2.0 * q4 + 2.0 * q5) + 0.00016599999 * sin(2.0 * q5)) + 0.5 * dq3 * (0.000165999 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00689 * sin(q2 + q3 + q4) + 0.0001659999 * sin(q2 + q3 + q4 + 2.0 * q5)) + 0.5 * dq4 * (0.0001659999 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.00689 * sin(q2 + q3 + q4) + 0.00016599999 * sin(q2 + q3 + q4 + 2.0 * q5))$$

$$C_{52} = 0.00008 * dq1 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.0001659999 * dq3 * sin(2.0 * q5) - 0.0001659999 * dq4 * sin(2.0 * q5) - 0.003445 * dq1 * sin(q2 + q3 + q4) - 0.0001659999 * dq2 * sin(2.0 * q5) + 0.0000829999 * dq1 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{53} = 0.0000829999 * dq1 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.000165999 * dq3 * sin(2.0 * q5) - 0.0001659999 * dq4 * sin(2.0 * q5) - 0.003445 * dq1 * sin(q2 + q3 + q4) - 0.00016599 * dq2 * sin(2.0 * q5) + 0.0000829999 * dq1 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{54} = 0.000082999 * dq1 * sin(q2 + q3 + q4 - 2.0 * q5) - 0.000165999 * dq3 * sin(2.0 * q5) - 0.000165999 * dq4 * sin(2.0 * q5) - 0.003445 * dq1 * sin(q2 + q3 + q4) - 0.000165999 * dq2 * sin(2.0 * q5) + 0.00008299999 * dq1 * sin(q2 + q3 + q4 + 2.0 * q5)$$

$$C_{55} = 0$$

- **the elements of the matrix $G(q)$:**

$G_{11} = 0$

$$G_{21} = 0.50624505 * sin(q2 + q3 + q4) - 4.913479 * cos(q2 - 1.53131) - 2.091396 * cos(q2 + q3 - 1.15872725)$$

$$G_{31} = 0.50624505 * sin(q2 + q3 + q4) - 2.09139 * cos(q2 + q3 - 1.1587)$$

$G_{41} = 0.50624505 * sin(q2 + q3 + q4)$

$G_{51} = 0$

Appendix (2):

GENERAL CHARACTERISTICS KUKA YOUBOT ARM:[4]

Serial kinematics	5 axes
Height	655 mm
Work envelope	0.513 m3
Weight	6.3 kg
Payload	0.5 kg
Structure	Magnesium cast
Positioning repeatability	1 mm
Communication Ether-CAT Voltage connection	24 V
Drive train power limitable to	80 W
Axis data	Range Speed
Axis 1(A1)	+/– 169° 90°/s
Axis 2 (A2)	+ 90°/– 65° 90°/s
Axis 3 (A3)	+ 146°/– 151° 90°/s
Axis 4 (A4)	+/– 102° 90°/s
Axis 5 (A5)	+/– 167° 90°/s
Gripper	Detachable, 2 fingers
Gripper stroke	20 mm
Gripper range	70 mm

GENERAL CHARACTERISTICS KUKA YOUBOT PLATFORM: [4]

Omnidirectional kinematics	4 KUKA omni-Wheels
Length	580 mm
Width	380 mm
Height	140 mm
Clearance	20 mm
Weight	20 kg
Payload	20 kg
Structure Steel Speed	0.8 m/s
Communication Ether-CAT Voltage connection	24 V

Energy supply maintenance-free lead acid rechargeable batteries: 24 V, 5 Ah, 4 kg Approximate runtime of youBot mobile manipulator: 90 minutes power adapter: 200 W, 24 V. [4]