COPPE
Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia
UFRJ

MANIPULABILITY IN TRAJECTORY TRACKING FOR CONSTRAINED REDUNDANT MANIPULATORS VIA SEQUENTIAL QUADRATIC PROGRAMMING

Felipe Figueiredo Cardoso

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Fernando Cesar Lizarralde

Rio de Janeiro
Julho de 2019

MANIPULABILITY IN TRAJECTORY TRACKING FOR CONSTRAINED REDUNDANT MANIPULATORS VIA SEQUENTIAL QUADRATIC PROGRAMMING

Felipe Figueiredo Cardoso

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

____
Prof. Fernando Cesar Lizarralde, D.Sc.

____
Prof. Alessandro Jacoud Peixoto, D.Sc.

____
Prof. Antonio Candea Leite, D.Sc.

____
Prof. Anna Carla Monteiro de Araujo, D.Sc.

____
Prof. Adriano Almeida Gonçalves Siqueira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2019

Cardoso, Felipe Figueiredo

Manipulability in Trajectory Tracking for Constrained Redundant Manipulators via Sequential Quadratic Programming/Felipe Figueiredo Cardoso. – Rio de Janeiro: UFRJ/COPPE, 2019.

XVI, 117 p.: il.; 29, 7cm.

Orientador: Fernando Cesar Lizarralde

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2019.

Referências Bibliográficas: p. 107 – 114.

1. Trajectory Tracking. 2. Redundant Manipulators. 3. Holonomic Constraints. I. Lizarralde, Fernando Cesar. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

MANIPULABILIDADE NO RASTREAMENTO DE TRAJETÓRIA PARA MANIPULADORES REDUNDANTES RESTRITOS VIA PROGRAMAÇÃO SEQUENCIAL QUADRÁTICA

Felipe Figueiredo Cardoso

Julho/2019

Orientador: Fernando Cesar Lizarralde

Programa: Engenharia Elétrica

Métodos de rastreamento de trajetória para manipuladores redundantes restritos são apresentados nesta tese, onde o efetuador de um manipulador serial redundante tem que rastrear uma trajetória desejada enquanto alguns pontos em sua cadeia cinemática satisfazem uma ou mais restrições. Além disso, dois índices de manipulabilidade são levados em consideração a fim de otimizar a trajetória para evitar singularidades. O primeiro índice é definido em função do jacobiano geométrico do manipulador na configuração restrita. O segundo índice é baseado no Jacobiano restrito, o qual mapeia velocidades no espaço das juntas para a espaço da tarefa, levando em conta as restrições holonômicas. Três métodos para resolver o problema de rastreamento de trajetória são discutidos. Os dois primeiros, controle cinemático e programação quadrática (QP), são amplamente discutidos na literatura. O terceiro, programação quadrática sequencial (SQP), é uma nova abordagem, diferentemente do controle cinemático ou QP, tem como vantagens (apesar de algumas deficiências) não depender explicitamente da pseudo-inversa de jacobianos, derivadas da trajetória desejada e linearização de índices ou restrições. Uma discussão desses três métodos é apresentada em termos de erro de rastreamento, violação da restrição, distância de singularidades, entre outros através de experimentos realizados em um robô colaborativo Baxter.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

MANIPULABILITY IN TRAJECTORY TRACKING FOR CONSTRAINED REDUNDANT MANIPULATORS VIA SEQUENTIAL QUADRATIC PROGRAMMING

Felipe Figueiredo Cardoso

July/2019

Advisor: Fernando Cesar Lizarralde

Department: Electrical Engineering

Trajectory tracking methods for constrained redundant manipulators are presented in this thesis, where the end-effector of a redundant serial manipulator has to track a desired trajectory while some points on its kinematic chain satisfy one or more constraints. In addition, two manipulability indexes are taken into account in order to optimize the trajectory. The first index is defined in terms of the geometric Jacobian of the manipulator in the constrained configuration. The second index is based on the constrained Jacobian, which maps velocities from joint space to task space, taking into account the holonomic constraints. Three methods for solving the trajectory tracking problem are discussed. The first two, kinematic control (KC) and quadratic programming (QP), are widely discussed in literature. The third, sequential quadratic programming (SQP), is a new approach, unlike KC or QP, has as advantages (despite some shortcomings) not explicitly depend on pseudo-inverse Jacobian, derivative from the desired trajectory and linearization of indexes or constraints. A discussion of these three methods is presented in terms of tracking error, constraint violation, singularity distance, among others through experiments performed on a Baxter collaborative robot.

Contents

List of Figures

List of Tables

List of Symbols

B	Body frame, p. 11
D	Matrix that defines constraint behavior, p. 22
F	Frame, p. 10
G	Inertial frame, p. 11
$H(\cdot)$	Hessian of a second order differentiable function, p. 35
I_i	Identity matrix of size i, p. 10
$J(\theta)$	Geometrical Jacobian, p. 18
K_p	Controller gain matrix, p. 21
P	Point in Cartesian space, p. 11
$R(r_p, \phi_p)$	Rotation matrix, p. 12
S	Selection matrix, p. 62
T	Sampling Period, p. 52
$T_{i,j}$	Homogeneous matrix from F_i to F_j, p. 13
V	Velocity, p. 18
W	Integral of $w(\theta)$, p. 79
Φ	Adjoint Matrix, p. 22
$\Psi(u)$	Merit function, p. 41
α	Weight in multi objective optimization, p. 51
γ_k	Step length of SQP, p. 40
λ	Lagrange multiplier, p. 38

$\mathbb{N}$	Natural numbers set, p. 10
$\mathbb{R}$	Real numbers set, p. 10
$\mathcal{E}$	Set of equality constraints, p. 34
$\mathcal{F}$	Set of optimization problem, p. 34
$\mathcal{I}$	Set of inequality constraints, p. 34
$\mathcal{L}(\cdot, \lambda)$	Lagrangian of a optimization problem, p. 38
$\mathcal{M}$	Set of objective functions, p. 50
$\mathcal{P}$	Pareto set, p. 51
$\mathcal{U}$	Set of velocity command, p. 34
$\mathsf{N}(\cdot)$	Null space, p. 24
$\nabla f(\cdot)$	Gradient of a differentiable function, p. 35
ω	Angular velocity, p. 11
ϕ_p	Rotation angle, p. 12
θ	Joint angle vector, p. 11
d_k	Search direction of SQP, p. 40
$det()$	Determinant of a matrix, p. 10
$diag()$	Diagonal matrix, p. 70
e	Error signal, p. 19
$f(\theta)$	Forward Kinematic function, p. 16
$g(u)$	Constraint function, p. 38
h	Axis of rotation, p. 18
m	Number of independent holonomic constraints, p. 21
n	Number of joints, p. 14
p	Pose, p. 20
r	Position vector, p. 11

$tr()$	Trace of a matrix, p. 10
u	Velocity command, p. 19
v	Linear velocity, p. 11
$w(\theta)$	Manipulability, p. 25
x	x axis, p. 10
y	y axis, p. 10
z	z axis, p. 10

List of Abbreviations

BFGS	Broyden-Fletcher-Goldfarb-Shanno, p. 46
IAE	Integral of the Absolute Error, p. 63
IK	Inverse Kinematics, p. 4
IPM	Interior Point Method, p. 43
ISE	Integral of the Square Error, p. 62
ITAE	Integral of the Time Multiplied by Absolute Error, p. 63
ITSE	Integral of Time Multiplied by the Squared Error, p. 63
KC	Kinematic Control, p. 5
LPD	Least Distance, p. 48
LSEI	Linear Least Squares, p. 46
LSI	Least Square, p. 48
NNLS	Non Negative Least Squares, p. 47
PID	Proportional Integral Derivative, p. 5
QP	Quadratic Programming, p. 5
ROS	Robot Operating System, p. 63
RPY	Roll-Pitch-Yaw angles, p. 14
SDK	Software Development System, p. 64
SEA	Series Elastic Actuator, p. 70
SLSQP	Sequential Least Squares Quadratic Programming, p. 37
SO(3)	Special group orthonormal of dimension 3, p. 10

SQP	Sequential Quadratic Programming, p. 6
URDF	Universal Robotic Description Format, p. 64
XML	Extensible Markup Language, p. 65

Chapter 1

Introduction

Robots play a key role nowadays performing tasks nearly impossible for humans, and the range of applications extends through aerospace, military, oil and gas industry, assembly lines, agriculture, medicine, among several others. There are several ways to classify robots according to their characteristics and one important definition is the redundant robot [15] - A redundant robot possesses more degree of freedoms than those strictly required to execute its task. This provides the robot with an increased level of dexterity that may be used to avoid singularities, joint limits, and workspace obstacles, but also to minimize joint torque, energy or, in general, to optimize suitable performance indexes.

In many situations the environment puts constraints that can be overcome by redundant robots, and in these situations robots need to satisfy constraints while performing tasks. Examples can be found in:

- Minimally invasive surgery, [25]: At the point of insertion of the surgical instrument into the patient's body during minimally invasive surgery the redundancy is used so the instrument does not move transversely in order to prevent tissue damage, as shown in Figure 1.1.

Figure 1.1: daVinci surgery system and insertion scheme in tissue, courtesy from Intuitive Surgical and adapted from [16], respectively.

- Decommissioning of oil production platforms, [31]: The additional degree of freedom is used to shape the effector's impedance in the task of biofouling scraping in submarine stakes, as illustrated in Figure 1.2.

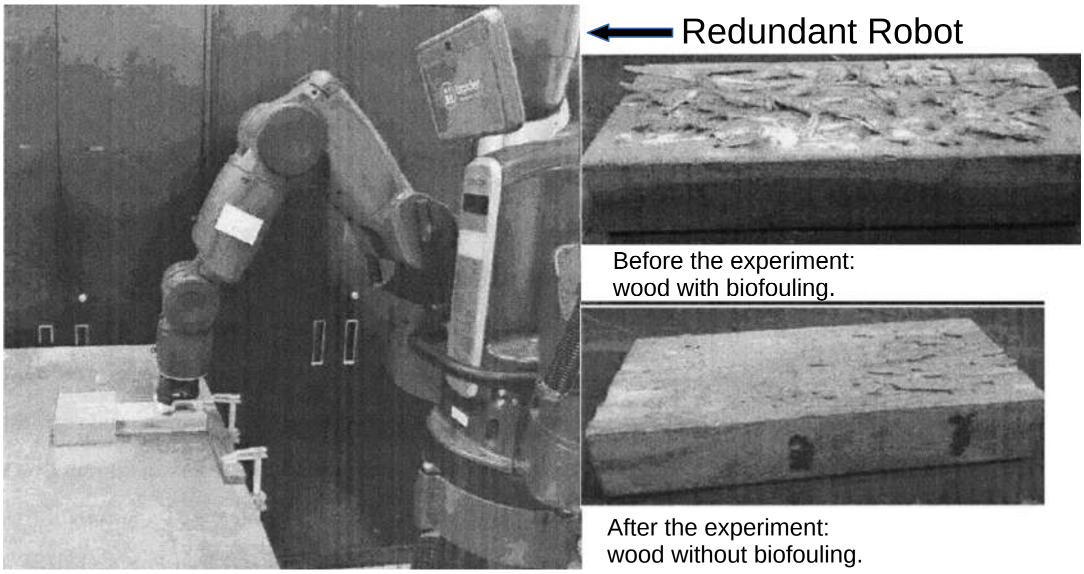


Figure 1.2: Biofouling removal experimental setup, from [31].

- Mapping of opaque pipeline geometry, [22]: A multi-segment snake robot maps the two-dimensional geometry of a pipeline through the joint angle sensors, Figure 1.3.

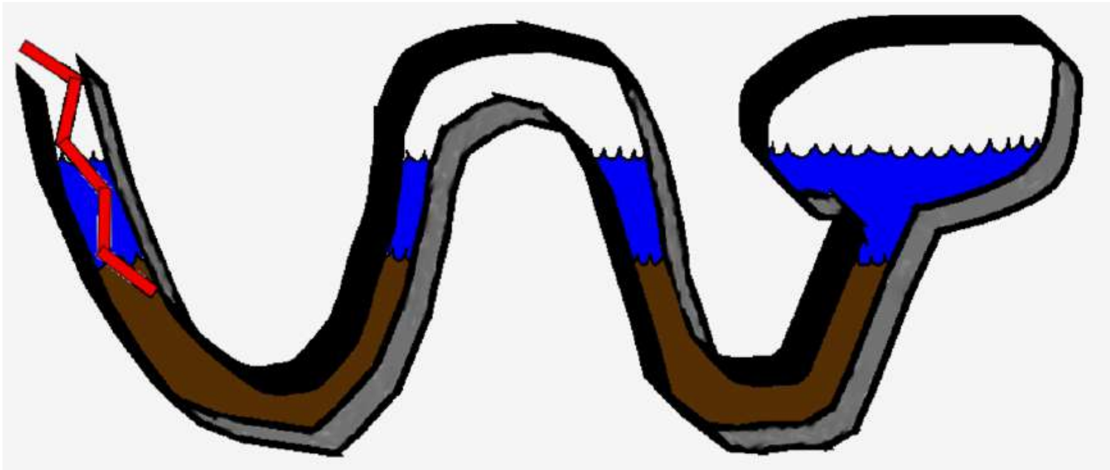


Figure 1.3: Snake robot exploring pipeline with different fluid conditions, from [22].

- Finite-time trajectory generation, [27]: A mobile platform gives a holonomic manipulator additional degrees of freedom to avoid obstacles in a trajectory generated in finite time, example in Figure 1.4.

Figure 1.4: Jaguar V4 platform with manipulator arm, courtesy from iRobotec.

1.1 Trajectory Tracking and Manipulability

A robot manipulator is defined as, adapted from [69] and [34] - A sequence or rigid bodies (links) interconnected by articulations (joints) to form a kinematic chain, along the chain there can be several devices needed for the manipulator to accomplish the desired tasks: actuators, sensors, end-effector, micro controllers, embedding processors, buttons among others. Figure 1.5 shows some manipulators examples.

The trajectory tracking problem for a robot manipulator is defined as, adapted from [69] - The end-effector asymptotically tracks a desired Cartesian trajectory starting from an initial configuration that may or may not be matched with the trajectory having the time as a constraint, so the end-effector have to be at a certain

Figure 1.5: Manipulators suited for specific tasks, computer numeric control and arc welding, courtesy from RobotWorx and Fanuc, respectively.

point at a certain time. On the other hand in the path following problem the end-effector follows a predefined path which does not involve time as a constraint, in this way the path can be followed at any velocity.

A range of diverse controllers have been proposed to solve the trajectory tracking problem in manipulators: the classical feedback controller in [71]; fuzzy logic control [44] for manipulators with uncertain kinematics and dynamics parameters; sliding mode control [9] using a hybrid position/force control scheme. As this work deals only with the kinematic equations, the chosen control scheme when using controllers is the classic feedback in kinematic approach.

In robotics manipulation a singularity according [72] - Represent configurations from which certain directions of motion may be unattainable where bounded end-effector velocities may correspond to unbounded joint velocities. Also, according [72] - When the manipulator approaches the singularity there will not exist a unique solution to the inverse kinematics (IK) problem, in such cases there may be no solution or there may be infinitely many solutions.

The manipulability measure of a robot according [54] is - The ability to change the position or orientation at a given configuration. The manipulability index can be defined (many different indexes have been proposed in the literature) by the product of singular values of the product of the Jacobian and its transpose. It may be noted that during a singular manipulator configuration the determinant of the Jacobian matrix is null, which means null singular values and also a null manipulability. So a manipulability analysis could help to improve a control strategy when redundant manipulators satisfy constraints because it is an indication of how close the manipulator is from a singular configuration.

In [26] is presented a performance metric for the manipulability of constrained serial manipulators that defines the constrained Jacobian matrix as an analytical mapping between the end-effector and joint velocities that also takes the kinematic chain constraints into account. [63] presents a task space control architecture when the manipulator chain satisfies a holonomic constraint while tracking a trajectory. This is done through a new set of velocity variables defined using the constrained Jacobian matrix, the trajectory error and a proportional plus feed-forward controller.

In recent years, with the rise of processing power, many optimization algorithms have been proposed to replace controllers in diverse situations. When it comes to trajectory tracking in manipulators, the optimization method named quadratic programming is presented by various authors as a viable alternative. The following works discussed in the next paragraph were chosen (because the completeness of description or using the same experimental setup) to show how quadratic programming is used in the trajectory tracking problem.

In [85] repetitive and nonrepetitive motion planning schemes are solved for redundant manipulator through formulation of a quadratic programming (QP) where neural networks and numerical algorithms QP solvers are used for finding the solutions of the IK problem. In [84] the work of [85] is expanded to include the maximization of a manipulability index. In [21] the IK problem is again solved through the QP, in addition to the index of manipulation also includes obstacle avoidance. In [43] the QP formulation of [85] is modified to include a proportional-integral-derivative (PID) term in order to have noise suppression capability.

1.2 Motivation

The classical feedback controller in kinematic approach (called from now on simply kinematic control or KC in some occasions) is a proven method for trajectory control with a stability proof. In redundant manipulators to minimize or maximize performance indexes, as example the manipulability, the kinematic control expands the null space of the manipulator Jacobian, this means that the analytical gradient of the performance index must be available [68], which is non practical, for example, when dealing with manipulability in manipulators with many joints. When dealing with constraints in the kinematic chain, only constraints that are linear to joint velocities fit the formulation in [63], so nonlinear constraints can not be considered.

The QP has an advantage over the kinematic control, the ease of defining equality and inequality constraints in the formulation of the optimization problem, also there is no need to adjust controllers parameters (except in some formulations) and the sampling period is not necessarily needed. However, it has some flaws too, as the kinematic control the constraints are all linear in terms of joint velocities(the system nonlinear constraints must be linearized in order to fit the QP formulation) and the objective function is quadratic at most.

1.3 Objectives

To overcome the main difficulties of kinematic control and QP, this thesis proposes to apply a constrained nonlinear optimization method in the trajectory tracking problem, considering that the redundant manipulator satisfy a holonomic constraint in its kinematic chain and also stay away from singular configurations through the maximization of the manipulability. There are many methods for solving nonlinear constrained optimization problems and as will be seen in Chapter 3.3 the sequential quadratic programming (SQP) is the designed method to solve the trajectory tracking problem for redundant manipulators including holonomic constraints and maximization of the manipulability in formulation.

Some advantages of SQP over kinematic control are: this approach does not depend explicitly on the pseudo-inverse of Jacobians, does not need the derivative of the desired trajectory, ease of defining equality and inequality nonlinear constraints in the formulation of optimization problem and there is no need to adjust controllers parameters. The advantages of SQP over QP are the possibility to define nonlinear constraints and objective function, also having the negative feedback loop for trajectory error.

The SQP shortcomings, in relation to IK and QP, are the convergence time and the lack of a formal test that guarantees the global stability of the method. The kinematic control method of [63] (including maximization of the manipulability) and QP [21, 43, 84, 85] (including holonomic constraints) are modified in a way that all three methods (kinematic control, QP and SQP) has the same framework: trajectory tracking for a redundant manipulator satisfying a holonomic constraint and maximizing the manipulability.

1.4 Contributions

The main contribution of this work is compare the SQP method with the traditional methods commonly found in literature, kinematic control and QP, for the trajectory tracking problem. Experiments are performed on a Baxter collaborative robot and a comparative table shows the differences among the methods through various aspects followed by a discussion about the ease of implementation and effectiveness.

The Table 1.1 is a comparison among some features of papers in kinematic control [63] and QP [21, 43, 84, 85], and also, the proposed SQP in this thesis. The primary objective is the trajectory tracking, so, all methods have this feature.

The maximization of manipulability is considering in [21] using approximations of gradient and Hessian of the manipulability function, in the proposed SQP there is no need for any approximation, then any complex geometry in the manipulability function is never discarded.

Holonomic constraints are dealt in [63] using the so called constrained Jacobian, a formulation that needs a methodology to be derived and needs to redo all calculations when the constraint is placed on another robot link or the constraint type is changed. Dealing with holonomic constraints in the proposed SQP is much simpler just insert an equation into the constraints of the optimization problem formulation.

Unlike in [21], the proposed SQP does not deal with distance to object. Only in [63] and [43] there is a need to adjust controller parameters, in case multiple holonomic constraints in the robot kinematic chain can result in a large number of parameters to be set.

Only the proposed SQP does not need to linearize or curve fitting functions because the objective function and constraints of the optimization problem may be nonlinear. Also, the SQP does not need the derivative of the trajectory tracking, thus, there is no need for inference from the derivative of a desired trajectory definite on the fly.

The QP and SQP methods are easily scalable, just add more constraints on the optimization problem formulation. Only the kinematic control in [63] has a global stability proof through Lyapunov. In terms of convergence time the proposed SQP is the slower, adding too many decisions variables or constraints may can make the problem unfeasible for online resolution.

The negative feedback loop for trajectory error in [21, 43] and the proposed SQP ensures that the desired trajectory is reached even in the presence of noise or system uncertainties.

Table 1.1: Comparison among methods for trajectory tracking.

Feature	KC [63]	QP [85]	QP [84]	QP [21]	QP [43]	SQP
Trajectory tracking	○	○	○	○	○	○
Maximizing manipulability			○	○		○
Holonomic constraint	○					○
Distance to object				○		
Need adjust controller parameters	○				○	
Need linearization		○	○	○	○	
Need curve fitting	○					
Need derivative of trajectory tracking	○	○	○	○	○	
Easily scalable		○	○	○	○	○
Global stability	○					
Negative feedback loop for trajectory error	○				○	○
Fast convergence	○	○	○	○	○	

1.5 Organization

This work is organized as follows:

- Chapter 2 - Introduces some fundamentals in kinematic theory, namely: geometric kinematics, derivative kinematics, kinematic control and constraints in applied mechanics.
- Chapter 3 - Presents the methods used for trajectory tracking: kinematic control, QP and SQP. Also, summarizes the implementation differences of trajectory tracking methods used.
- Chapter 4 - Experiments and simulation are performed using kinematic control, QP and SQP methods.
- Chapter 5 - Closes the thesis with the conclusions and future works.

Chapter 2

Robot Kinematics

In this section topics of geometric kinematics are discussed. It presents the Cartesian coordinate system, rotation kinematics, motion kinematics forward kinematics and constraints. The text and the figures are mainly adapted from [34].

2.1 Geometric Kinematics

Geometric kinematics is the branch of science that studies geometry in motion but does not take its causes into account. Motion means any kind of displacement that implies a change in position or orientation. The orthogonality conditions (property in which the scalar product of vectors is null) and geometric transformation (correspondence between points of the same or different spaces) are the basis of geometric kinematics [34].

2.1.1 Notation

The following notation and definitions are used throughout the thesis: $\mathbb{R} := (-\infty, \infty)$, $\mathbb{R}^+ := [0, \infty)$ and $\mathbb{N}$ the natural numbers set. A superscript T denotes the transpose, matrix or vector. $det()$ is the determinant and $tr()$ the trace of a matrix. I_i is the identity matrix of size i. x , y and z are the axes of a Euclidean space formed by the canonical unit vectors, i.e. $x_c = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$, $y_c = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$ and $z_c = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, respectively. A subscript x, y or z means with respect to x axis, y axis or z axis, respectively. $0_{i,j}$ is a all zero matrix of i lines and j columns. An upper dot $(\dot{\ })$ is the time-derivative. An upper hat $(\hat{\ })$ is the skew symmetric matrix. $SO(3)$ is the special orthogonal group of dimension 3, matrix $R \in \mathbb{R}^{3\times 3}$ for example, $SO(3) = \left\{R \in \mathbb{R}^{3\times 3},\ R^T R = I_3 \text{ and } det(R) = 1\right\}$. $t \in \mathbb{R}$ is the time.

A frame is represented by F. The subscript i means a reference for the i-th element. A subscript (i, j) in a matrix or vector denotes the matrix or vector from

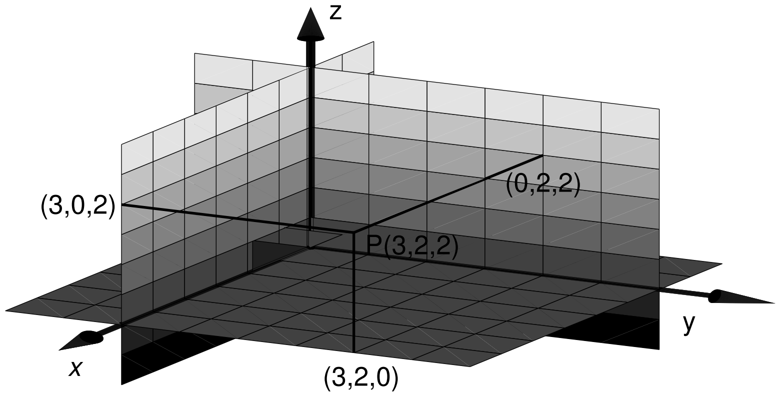


Figure 2.1: Cartesian coordinate system, adapted from [35]

F_i to F_j. The total numbers of joints is $n \in \mathbb{N}$. The joint angle vector, a generalized coordinate definite in joint space, is represented by $\theta \in \mathbb{R}^n$, a joint angle in the frame i is denoted by θ_i, a joint angle vector between F_i and F_j is represented by $\theta_{i,j} = [\theta_i \; \theta_{i+1} \; \cdots \; \theta_{j-1} \; \theta_j]^T$. The linear and angular velocities are denoted by $v \in \mathbb{R}^3$ and $\omega \in \mathbb{R}^3$, respectively. The subscript G means the variable is defined in the inertial frame while the subscript B means the variable is defined in the body frame.

2.1.2 Cartesian Coordinate System

A Cartesian coordinate system is settled from a set of parallel and mutually perpendicular planes. The intersection of a pair of planes defines an axis and the three intercepting axes define a base, Figure 2.1. The axes x-axis, y-axis and z-axis are perpendicular to the planes x-plane, y-plane and z-plane, respectively.

The position of a point in the Cartesian space is an intersection of three planes. In Figure 2.1, for example, the point $P \in \mathbb{R}^3$ is the intersection of three planes parallel to y-z-planes (distance $x = 3$), z-x-planes (distance $y = 4$) and x-y-planes (distance $z = 2$), so the Cartesian coordinates are $P(x, y, z) = P(3, 4, 2)$.

The vectors of position, $r \in \mathbb{R}^3$, and linear velocity of a point P moving in the Cartesian space are defined by (2.1) and (2.2), respectively:

$$r = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}, \tag{2.1}$$

$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \dot{r} = \begin{bmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{r}_z \end{bmatrix}. \tag{2.2}$$

2.1.3 Rotation Kinematics

Consider a rigid body in a frame B (body frame) that is originally coincident with the inertial frame G. Figure 2.2 shows a position vector in red color representing the rigid body at the same position in the frames G and B.

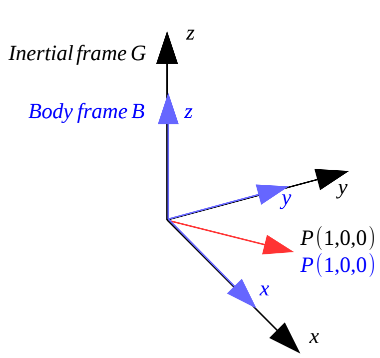


Figure 2.2: Position vector in red color at coincident inertial and body frames, inertial frame in black color and body frame in blue color. The point P have equal values in the frames.

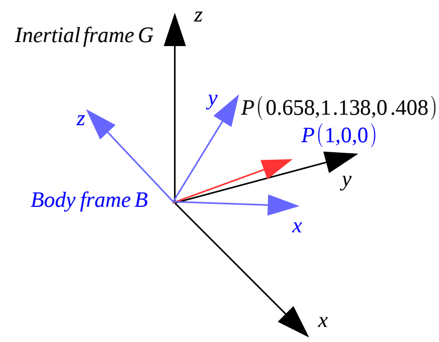


Figure 2.3: Position vector in color red after rotating. The value of the point P changes in the inertial frame and remains unchanged in the body frame. $R(r_p, \phi_p)$ is defined by $r_p = [0.775\ 0.447\ 0.447]^T$ and $\phi_p = 0.813\ rad$.

Consider that the position vector in Figure 2.2 rotates about a fixed vector. As a result the inertial frame G and the body frame B are not more coincident as seen in Figure 2.3. In this way the point P have different values for inertial and body frames and the position vector is related by:

$$r_G = R_{G,B}(r_p, \phi_p) r_B, \tag{2.3}$$

where $R_{G,B}(r_p, \phi_p) \in SO(3)$ is a rotation matrix between the body and inertial frames described through a rotation by a $\phi_p \in \mathbb{R}$ angle around a fixed vector $r_p \in \mathbb{R}^3$.

Considering that r_p is a unit vector, the rotation matrix in (2.3) is given by the Euler's rotation theorem:

$$R_{G,B}(r_p, \phi_p) = e^{\hat{r}_p \phi_p}, \tag{2.4}$$

where $\hat{r}_p \in \mathbb{R}^{3\times3}$ is the skew symmetric matrix of the unit vector r_p. The term $e^{\hat{r}_p\phi_p}$ is given by the Rodrigues's rotation formula [34]:

$$e^{\hat{r}_p\phi_p} = I_3 + \sin(\phi_p)\hat{r}_p + (1 - \cos(\phi_p))\hat{r}_p^2. \tag{2.5}$$

2.1.4 Motion Kinematics

Consider again a rigid body in a frame B that is originally coincident with the inertial frame G. A position vector in red color representing the rigid body at the same position in the frames G and B can be seen in Figure 2.4.

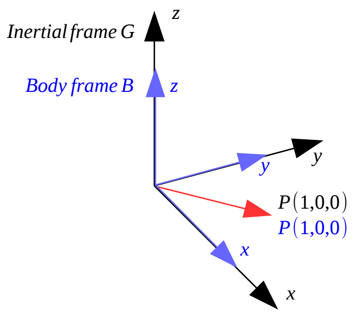


Figure 2.4: Position vector in red color at coincident inertial and body frames, inertial frame in black color and body frame in blue color. The point P have equal values in the frames.

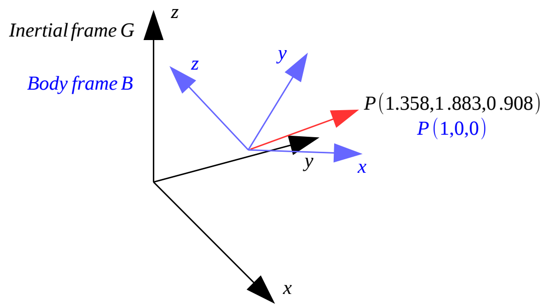


Figure 2.5: Position vector in color red after rotating and translating. The value of the point P changes in the inertial frame and remains unchanged in the body frame. $R(r_p, \phi_p)$ is defined by $r_p = [0.775\ 0.447\ 0.447]^T$ and $\phi_p = 0.813\ rad$ while $r_{G,B} = [0.7\ 0.7\ 0.5]^T$.

Consider now that the position vector also translates besides rotating about a fixed vector. Again, as a result, the inertial frame G and the body frame B are not more coincident, now in position and orientation at Figure 2.5.

In order to facilitate calculations, the position vector in (2.1) can be represented by a $(3 + 1)$-component vector, where the appended element is a scale factor that will be equal to 1, called homogeneous position vector, $\begin{bmatrix} r & 1 \end{bmatrix}^T \in \mathbb{R}^4$. The homogeneous position vector at inertial and body frames is related by:

$$\begin{bmatrix} r_G \\ 1 \end{bmatrix} = T_{G,B} \begin{bmatrix} r_B \\ 1 \end{bmatrix}, \tag{2.6}$$

where $T_{G,B} \in \mathbb{R}^{4\times4}$ maps a rotation (through $R_{G,B}(r_p, \phi_p)$) and a translation (through the distance from frame G to frame B given by $r_{G,B} \in \mathbb{R}^3$) between frames

G and B:

$$T_{G,B} = \begin{bmatrix} R_{G,B}(r_p, \phi_p) & r_{G,B} \\ 0_{1,3} & 1 \end{bmatrix}. \tag{2.7}$$

2.1.5 Multibody

A multibody is a mechanical system connected by two or more rigid bodies. Each member of the multibody that can move relative to all other members is a link. The links are connected by joints. A revolute joint, as shown in Figure 2.6, allows relative rotation between two joints. Without loss of generality, here in this work, only revolute joints are considered.

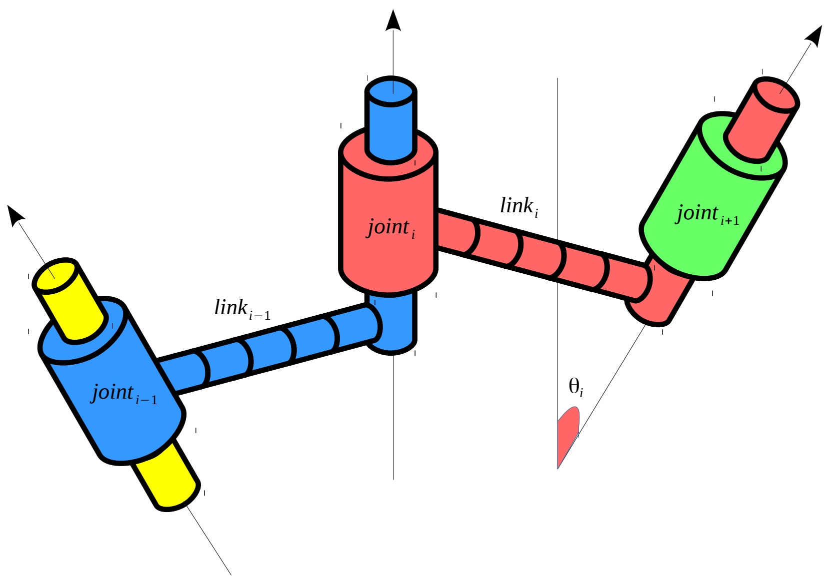


Figure 2.6: Revolute joint, θ_i is connecting the $i-1$-th link and the i-th link.

A serial multibody with n revolute joints, also called serial manipulator, can be seen in Figure 2.7. The inertial frame is F_0, frame F_i $(i = 1, \ldots, n)$ is the frame associated with the i-th link and F_e is the end-effector frame. A joint angle, generalized coordinate, in F_i is denoted by θ_i, where each θ_i is related with a revolute joint.

2.1.6 Roll-Pitch-Yaw Angles

The rotation matrix of (2.4) needs nine parameters to represent orientation, this way it has a very limited use in control due to the difficulty of handling all nine elements [8]. A alternative option for a more compact representation is use the Roll-Pitch-Yaw (RPY) angles which have three independent parameters. These parameters are

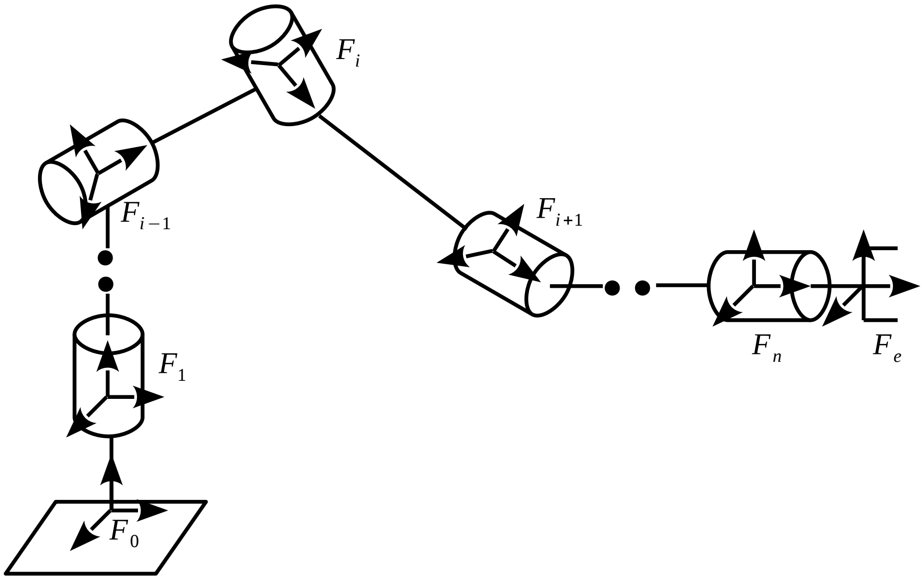


Figure 2.7: Serial manipulator with revolute joints.

the ϕ_x roll angle (rotation in x axis), the ϕ_y pitch angle (rotation in y axis) and the ϕ_z yaw angle (rotation in z axis).

The RPY is defined by consecutive rotations around the x, y and z. Thus, a coordinate transformation between two frames is defined by:

$$R(\phi) = R(z_c, \phi_z)R(y_c, \phi_y)R(x_c, \phi_x). \tag{2.8}$$

where $\phi = \begin{bmatrix} \phi_x & \phi_y & \phi_z \end{bmatrix}^T \in \mathbb{R}^3$ is the manipulator orientation.

To represent a rotation matrix by RPY angles the following expression is defined:

$$R(\phi)=\begin{bmatrix} \cos(\phi_y)\cos(\phi_z) & \sin(\phi_x)\sin(\phi_y)\cos(\phi_z)-\cos(\phi_x)\sin(\phi_z) & \cos(\phi_x)\sin(\phi_y)\cos(\phi_z)+\sin(\phi_x)\sin(\phi_z) \\ \cos(\phi_y)\sin(\phi_z) & \sin(\phi_x)\sin(\phi_y)\sin(\phi_z)+\cos(\phi_x)\cos(\phi_z) & \cos(\phi_x)\sin(\phi_y)\sin(\phi_z)-\sin(\phi_x)\cos(\phi_z) \\ -\sin(\phi_y) & \sin(\phi_x)\cos(\phi_y) & \cos(\phi_x)\cos(\phi_y) \end{bmatrix}. \tag{2.9}$$

2.1.7 Forward Kinematics

Forward kinematics is the transformation of kinematic information from joint space (joint angles values) to task space (position and orientation in Cartesian coordinates). In this way, the objective of forward kinematics is to determine the position and orientation of every frame in a multibody for a set of joint angles.

The orientation of a frame can be found through rotation matrices. In a manipulator chain the rotation matrix between two frames F_i and F_j is given by

$$R_{i,j}(r_p, \phi_p) = \prod_{i=1}^{i=j-1} R_{i,i+1}(r_{pi}, \phi_i), \tag{2.10}$$

where $R_{i,i+1}(r_{pi}, \phi_i) \in SO(3)$ is the rotation matrix between two consecutive frames. It is useful to define $R_{i,i+1}(r_{pi}, \phi_i) = e^{\hat{r}_{pi}\phi_i}$ where $\phi_i = \theta_i$ and for the sake of simplicity r_{pi} is always equal to one of the canonical unit vectors, i.e, x_c, y_c or z_c. For example if the link i rotates around z results $R_{i,i+1} = e^{\hat{z}_c\theta_i}$. The orientation of F_i with respect to the inertial frame is defined by $R_{0,i}(r_p, \phi_p)$ in (2.10).

The homogeneous transformation matrix, $T_{i,i+1} \in \mathbb{R}^{4\times4}$ that maps a position vector from F_i to F_{i+1} is:

$$T_{i,i+1} = \begin{bmatrix} R_{i,i+1}(r_{pi}, \phi_i) & (r_{i,i+1})_i \\ 0_{1,3} & 1 \end{bmatrix}, \tag{2.11}$$

where $(r_{i,i+1})_i \in \mathbb{R}^{3\times3}$ is the position vector from F_i to F_{i+1} represented in F_i.

In a manipulator chain the homogeneous matrix between F_i and F_j is

$$T_{i,j} = \prod_{i=1}^{i=j-1} T_{i,i+1}, \tag{2.12}$$

so the position of F_i to the inertial frame is defined using $T_{0,i}$ in (2.12):

$$\begin{bmatrix} r_0 \\ 1 \end{bmatrix} = T_{0,i} \begin{bmatrix} r_i \\ 1 \end{bmatrix}. \tag{2.13}$$

In a robot manipulator with only revolute joints the pose only depends on the joint angles (system variables):

$$p = \begin{bmatrix} r \\ \phi \end{bmatrix} = \mathsf{FK}(\theta), \tag{2.14}$$

where the pose $p \in \mathbb{R}^\eta$ is generally defined as the position plus orientation of a given part of the manipulator as example the end-effector (the pose can also be defined in terms of the task space variables), $\eta \in \mathbb{N}$ being the dimension of chosen representation with $\mathsf{FK}(\theta)\mathbb{R}^n \mapsto \mathbb{R}^\eta$ representing the forward kinematic function.

The $\mathsf{FK}(\theta)$ can be split into two parts, position and orientation. The position part can be found using (2.13). Considering orientation is given by RPY angles, $\eta = 3$ and $p \in \mathbb{R}^6$, so it can be found using (2.9) and (2.10), which leads the following relations:

$$\phi_x = atan2(R(\phi)_{3,2}, R(\phi)_{3,3}), \tag{2.15}$$

$$\phi_y = atan2(-R(\phi)_{3,1}, \sqrt{R(\phi)_{3,2}^2 + R(\phi)_{3,3}^2}), \tag{2.16}$$

$$\phi_z = atan2(R(\phi)_{2,1}, R(\phi)_{1,1}), \tag{2.17}$$

where $atan2(\cdot)$ is the two argument arctangent function and $R(\phi)_{i,j}$ is an element of $R(\phi)$ from the i-th row and j-th column.

2.2 Differential Kinematics

The differential kinematics defines the relationship between the joint velocities and accelerations to the corresponding manipulator velocities and accelerations on task space. This section presents the derivative kinematics dealing about velocity kinematics, geometric Jacobian and the relation between joint and task space. The text and the figures are mainly adapted from [35].

2.2.1 Velocity Kinematics

Consider again a rigid body represented by a position vector in a frame B that is originally coincident with the inertial frame G as shown in Figure 2.8. Consider now that the frame B is rotating with a angular velocity $\dot{\phi}_p$ about a vector r_p, Figure 2.9.

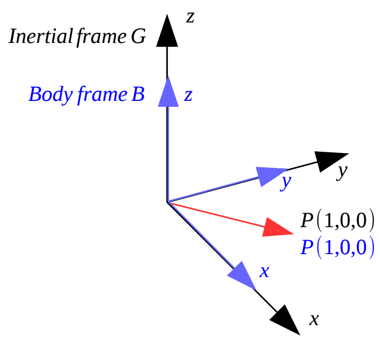


Figure 2.8: Position vector in red color at coincident inertial and body frames, inertial frame in black color and body frame in blue color. The point P have equal values in the frames.

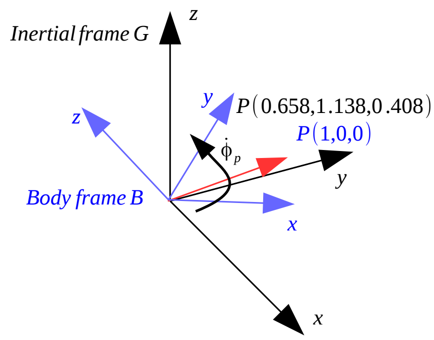


Figure 2.9: Position vector in color red after rotating at a angular velocity. The value of the point P changes in the inertial frame and remains unchanged in the body frame. $R(r_p, \dot{\phi}_p, t)$ is defined by $r_p = [0.775\ 0.447\ 0.447]^T$, $\dot{\phi}_p = 0.813\ rad/s$ and $t = 1\ s$ while $\omega = [0.630\ 0.363\ 0.363]^T$.

The coordinates in inertial frame of a position vector in rotation with constant velocity are:

$$r_G = R(r_p, \dot{\phi}_p, t) r_B = e^{\hat{r}_p \dot{\phi}_p t}, \tag{2.18}$$

The velocity of a position vector in the inertial frame F_0 is:

$$v_G = \dot{r}_G = \hat{\omega}_B r_G \tag{2.19}$$

where $\omega_B \in \mathbb{R}^3$ is the angular velocity vector of F_B. Considering a rotating frame with $R(r_p, \dot{\phi}_p, t)$, ω is defined by:

$$\omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \dot{\phi}_p r_p. \tag{2.20}$$

2.2.2 Geometric Jacobian

In robotics, the differential kinematics determines the linear velocities of the manipulator kinematic chain from the joint velocities, which are represented by the following vector $\dot{\theta} \in \mathbb{R}^n$:

$$\dot{\theta} = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dots & \dot{\theta}_{n-1} & \dot{\theta}_n \end{bmatrix}^T. \tag{2.21}$$

The velocity $V \in \mathbb{R}^6$ on a point in the kinematic chain has respectively the components of linear and angular velocities in the axes x, y and z:

$$V = \begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} v_x & v_y & v_z & \omega_x & \omega_y & \omega_z \end{bmatrix}^T. \tag{2.22}$$

The geometrical Jacobian $J(\theta) \in \mathbb{R}^{6\times n}$ maps the joint velocities to Cartesian velocities:

$$V = J(\theta)\dot{\theta}. \tag{2.23}$$

The geometric Jacobian for a point P in the manipulator kinematic chain after the joint n and before the joint $n+1$ is defined as (only revolute joints are considered):

$$J_P(\theta_{1,n}) = \begin{bmatrix} h_1 \times r_{1,P} & h_2 \times r_{2,P} & \cdots & h_{n-1} \times r_{n-1,P} & h_n \times r_{n,P} \\ h_1 & h_2 & \cdots & h_{n-1} & h_n \end{bmatrix}, \tag{2.24}$$

where $h_i \in \mathbb{R}^3$ is the axis of rotation of the joint i and $r_{i,P}$ is the displacement vector between the joint i and the point P. For a Jacobian matrix $J(\theta)$, $J^T(\theta)(J(\theta)J^T(\theta))^{-1}$ is the pseudo-inverse denoted by $J^\dagger(\theta)$.

2.2.3 Analytical Jacobian

The analytical Jacobian $J_A(\theta) \in \mathbb{R}^{\eta\times n}$relates the changes in robots joints angles to spatial velocity in the task space:

$$\dot{p} = J_A(\theta)\dot{\theta}, \tag{2.25}$$

where $\dot{p} \in \mathbb{R}^\eta$ is the time derivative of pose.

The expression of the analytical Jacobian is dependent of the parameterization and can be calculated using the partial derivative of the pose:

$$J_A(\theta) = \frac{\partial p}{\partial \theta}. \tag{2.26}$$

For computational purposes (2.26) is not very practical. So, it is useful to define the relation of ω and the time derivative of the orientation vector $\dot{\phi}$ given by:

$$\omega = J_R(\phi)\dot{\phi}, \tag{2.27}$$

considering RPY angles $J_R(\phi) \in \mathbb{R}^{3\times 3}$ is the representation Jacobian given by:

$$\begin{bmatrix} 1 & 0 & \sin(\phi_y) \\ 0 & \cos(\phi_x) & -\cos(\phi_y)\sin(\phi_x) \\ 0 & \sin(\phi_x) & \cos(\phi_y)\cos(\phi_x) \end{bmatrix} \tag{2.28}$$

That way a expression considering RPY angles for the analytical Jacobian $J_A(\theta) \in \mathbb{R}^{6\times 6}$ is given by:

$$J_A(\theta) = \begin{bmatrix} I_3 & 0_{3\times 3} \\ 0_{3\times 3} & J_R^{-1}(\phi) \end{bmatrix} J(\theta). \tag{2.29}$$

2.3 Kinematic Control

This section presents the kinematic control for a serial manipulator with n joints. There are two assumptions to be considered in a kinematic approach: first, the forward kinematics of the serial manipulator is known; second, the dynamic effects can be neglected because the tasks require only low joints speed and acceleration including the joints gear reductions ratios are elevated.

It is considered that the robot manipulator has an internal control loop for joint velocity, as shown by the block diagram in Figure 2.10. The velocity command $u \in \mathbb{R}^n$ is the reference signal while the error signal $e \in \mathbb{R}^n$ is:

$$e = u - \dot{\theta}. \tag{2.30}$$

Still in Figure 2.10 the controller is a pure proportional with a high gain that scales the error in order to generate the control signal $\nu \in \mathbb{R}^n$ sent to the driver. The torques $\tau \in \mathbb{R}^n$ generated by the driver are sent to the robot joint motors, responsible for the joint movements. With this internal control loop $e \to 0$ and $u \approx \dot{\theta}$.

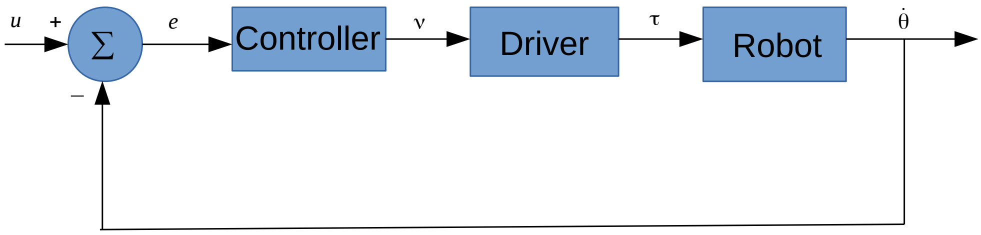


Figure 2.10: Internal joint velocity control loop.

2.3.1 Pose Control in Cartesian Space

The pose kinematic control in Cartesian space (considering RPY angles) means the end-effector pose $p_e \in \mathbb{R}^6$ tracks a desired time-varying trajectory $p_d(t) \in \mathbb{R}^6$, so in the ideal case $p_e \to p_d(t)$, .

The Figure 2.11 shows a block diagram for a kinematic control loop in Cartesian space and the block Internal control loop refers to the block diagram in Figure 2.10. Utilizing the analytical Jacobian until the end-effector $J_{Ae}(\theta) \in \mathbb{R}^{6 \times n}$ is possible to obtain the end-effector pose derivative $\dot{p}_e \in \mathbb{R}^m$:

$$\dot{p}_e = J_{Ae}(\theta)\dot{\theta}. \tag{2.31}$$

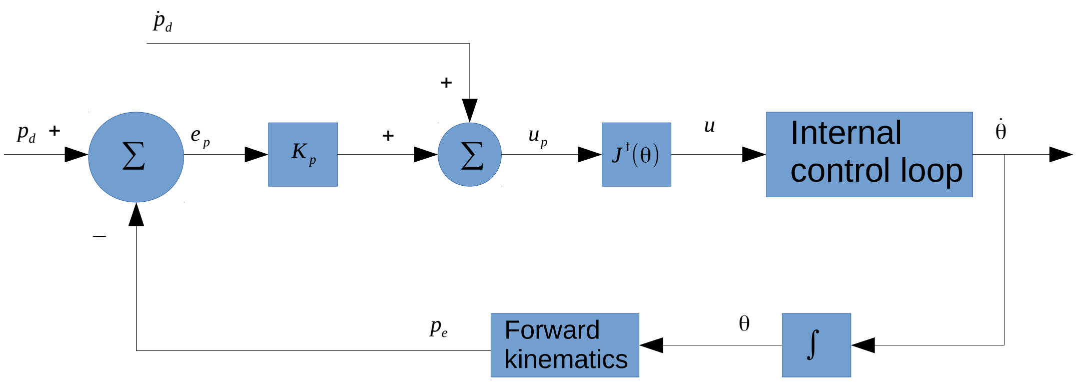


Figure 2.11: Kinematic position control loop.

Integrating $\dot{\theta}$ over time and applying the forward kinematics results in the end-effector pose p_e. Still, in Figure 2.11 the pose error $e_p \in \mathbb{R}^6$ is:

$$e_p = p_d(t) - p_e. \tag{2.32}$$

The proposed controller $u_p \in \mathbb{R}^6$ is a proportional plus feed forward, the proportional term $K_p \in \mathbb{R}^{6\times 6}$ is a diagonal gain matrix while the feed forward term $\dot{p}_d(t) \in \mathbb{R}^6$ is the derivative of the desired trajectory:

$$u_p = K_p e_p + \dot{p}_d(t). \tag{2.33}$$

In order to obtain u, the input for the internal control loop, it is used the Jacobian pseudoinverse $J^{\dagger}_{Ae}(\theta) \in \mathbb{R}^{n\times 6}$:

$$u = J^{\dagger}_{Ae}(\theta) u_p. \tag{2.34}$$

To obtain the position error dynamics derivative (2.32), substitutes $\dot{p}_d(t)$ with (2.33) and considering that $\dot{\theta} = u$ and substituting 2.34 in 2.31 implies $u_p = \dot{p}_e$:

$$\dot{e}_p = \dot{p}_d(t) - \dot{p}_e = u_p - K_p e_p - \dot{p}_e = -K_p e_p \tag{2.35}$$

where with a positive definite K_p matrix implies that $\lim_{t\to\infty} e_p(t) = 0$.

2.4 Constraints in Applied Mechanics

A constraint is defined as the limitation in motion of particles and rigid bodies. There are many classifications for constraints and only a subset is discussed in this section. For a more detailed and complete explanation see [35, 74].

A holonomic system is a system where it is possible express one coordinate in therms of others coordinates in equations that involves only position variables and time. As the pose in (2.14) defined by the forward kinematics only depends on θ, the system variables that describe position, a manipulator that conforms with $p = \mathsf{FK}(\theta)$ is a holonomic manipulator. A system where the pose is defined in terms of the derivatives of system variables is called a non-holonomic system.

For the holonomic manipulator all imposed constraints in the manipulator chain are also holonomic, being defined by equality equations in terms of positions variables (joint angles) or those that can be integrated to position level equations if initially described by velocity level equations [82].

Another classification of constraints which is independent of the constraint being holonomic or non-holonomic, as long the constraints are expressed in terms of systems variables [58], is the nomenclature scleronomic and rheonomic. A scleronomic (or stationary) constraint does not change as a function of time while a rheonomic (or non-stationary) constraint varies with time.

Regarding the holonomic manipulator with only revolute joints, a scleronomic constraint is defined as:

$$f(\theta) = v_d(t) = \text{constant}, \tag{2.36}$$

where $v_d(t) \in \mathbb{R}^m$ is the desired velocity of the point with $m \in \mathbb{N}$ being the number of independent holonomic constraints, and for a scleronomic constraint is mandatory that $v_d(t)$ is a constant. On the other hand, a rheonomic constraint is defined as:

$$f(\theta) = v_d(t), \tag{2.37}$$

with $v_d(t) \in \mathbb{R}^m$ being a desired velocity dependent of time.

2.5 Constrained Jacobian

Considering open chain serial manipulators the constrained Jacobian [63] is the matrix that maps velocities from joint space to task space when the manipulator satisfy holonomic constraints. The following procedure shows how to obtain the constrained Jacobian when the manipulator has holonomic constraints at only one point in chain. From now on, all velocities and Jacobians are considered in the body frame, the superscript notation with B is ignored to make equations cleaner.

In Figure 2.12 a constrained serial manipulator with n revolute joints is presented. As in Figure 2.7 F_0 is inertial frame, F_i $(i = 1, ..., n)$ the frame attached to the i-th joint, F_e the end-effector frame and each θ_i is related with the i-th revolute joint. Two new frames are defined in Figure 2.12, F_b the frame in the joint before the holonomic constraints and F_c the frame at the holonomic constraints. The velocity, $V_b \in \mathbb{R}^6$, at F_b and the joint velocity are related by:

$$V_b = J_b(\theta_{1,b})\dot{\theta}_{1,b}, \tag{2.38}$$

where $J_b(\theta_{1,b}) \in \mathbb{R}^{6\times b}$ is a partial Jacobian.

The velocity at F_c and F_b are related by:

$$V_c = \Phi_{c,b} V_b, \tag{2.39}$$

where the adjoint matrix $\Phi_{i,j} \in \mathbb{R}^{6\times 6}$ maps velocities between F_i and F_j:

$$\Phi_{i+1,i} = \begin{bmatrix} R^T_{i,i+1}(r_{pi}, \theta_i) & -R^T_{i,i+1}(r_{pi}, \theta_i)[(r_{i,i+1})_i]_\times \\ 0 & R^T_{i,i+1}(r_{pi}, \theta_i) \end{bmatrix}, \tag{2.40}$$

where $(r_{i,i+1})_i \in \mathbb{R}^3$ is the displacement vector between frames F_i and F_{i+1} represented in F_i.

The kinematic chain of the manipulator satisfy a holonomic constraint at F_c. Then it is considered that there is a scleronomic holonomic constraint at F_c defined using a matrix $D \in \mathbb{R}^{m\times 6}$ that defines constraint behavior where m is the dimension

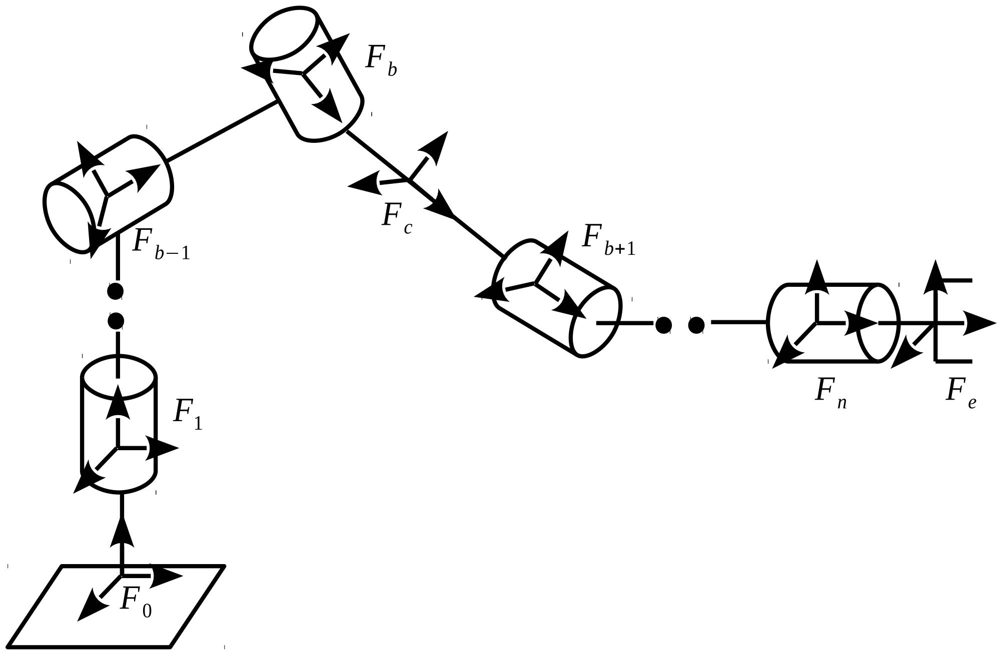


Figure 2.12: Constrained serial manipulator with revolute joints.

of the constraint and $v_d(t) \in \mathbb{R}^m$ is zero, i.e.,

$$DV_c = v_d(t) = 0, \tag{2.41}$$

while a rheonomic holonomic constraint at F_c is defined using a desired velocity $v_d(t) \in \mathbb{R}^m$ which is time dependent together with D, i.e.,

$$DV_c = v_d(t). \tag{2.42}$$

Each line in matrix D (considering that all lines are linearly independent) in (2.41) or (2.42) for simplicity have a Euclidean norm equal to one and defines only one holonomic constraint. The direction of a displacement constraint is defined using the first 3 columns (components is axes x, y and z, respectively) of a line. On the other hand a direction of a rotating constraint is defined using the last 3 columns (components is axes x, y and z, respectively) of a line. So, each row in D can define only one type of constraint, displacement or rotation.

In order to ensure task feasibility is mandatory that $m < b$, otherwise the number of degrees of freedom would be zero or negative implying that only self motion ($m = b$) or no motion at all ($m > b$) at frame F_c. Also, $m < 6$ is mandatory because when $m = 6$ the frame F_c can not move or rotate in any direction.

Substituting (2.38) and (2.39) in (2.41), one has

$$D\Phi_{c,b} J_b(\theta_{1,b})\dot{\theta}_{1,b} = 0. \tag{2.43}$$

The joint velocity vector satisfying (2.43) is given by:

$$\dot{\theta}_{1,b} = J_b{}^{\dagger}(\theta_{1,b})\mathsf{N}(D\Phi_{c,b})u_f, \tag{2.44}$$

where $\mathsf{N}(D\Phi_{c,b})$ spans the null space of $D\Phi_{c,b}$ and $u_f \in \mathbb{R}^{b-m}$ is a control degree of freedom. The dimension of u_f is defined using the manipulator degrees of freedom until the holonomic constraints minus the number of independent holonomic constraints.

On the other hand, the end-effector velocity is given by:

$$V_e = J_e(\theta)\dot{\theta}. \tag{2.45}$$

Partitioning $J_e(\theta)$ into two parts, the end-effector velocity can be written as:

$$V_e = \begin{bmatrix} J_{e1}(\theta) & J_{e2}(\theta_{b+1,n}) \end{bmatrix} \begin{bmatrix} \dot{\theta}_{1,b} \\ \dot{\theta}_{b+1,n} \end{bmatrix}. \tag{2.46}$$

Replacing (2.44) in (2.46) creates:

$$V_e = \begin{bmatrix} J_{e1}(\theta)J_b{}^{\dagger}(\theta_{1,b})\mathsf{N}(D\Phi_{c,b}) & J_{e2}(\theta_{b+1,n}) \end{bmatrix} \begin{bmatrix} u_f \\ \dot{\theta}_{b+1,n} \end{bmatrix}. \tag{2.47}$$

In (2.47) the matrix multiplication $J_{e1}(\theta)J_b{}^{\dagger}(\theta_{1,b})$ only depends on $\theta_{b+1,n}$ considering $J_b(\theta_{1,b})$ not singular, please, see the proof in [16]. Thus, $J_r(\theta_{b+1,n}) \in \mathbb{R}^{6\times n-m}$, called constrained Jacobian matrix, is defined:

$$J_r(\theta_{b+1,n}) = \begin{bmatrix} J_{e1}(\theta)J_b{}^{\dagger}(\theta_{1,b})\mathsf{N}(D\Phi_{c,b}) & J_{e2}(\theta_{b+1,n}) \end{bmatrix}. \tag{2.48}$$

Consolidating this methodology an algorithm is used to determine the constrained Jacobian, defined by Algorithm 1.

Algorithm 1 Constrained Jacobian algorithm.

Define D
Define $J_b(\theta_{1,b})$
Define $J_e(\theta)$
$J_{e1}(\theta) \leftarrow$ first b columns of $J_e(\theta)$
$J_{e2}(\theta_{b+1,n}) \leftarrow$ last $n-b$ columns of $J_e(\theta)$
$J_r(\theta_{b+1,n}) \leftarrow \begin{bmatrix} J_{e1}(\theta)J_b{}^{\dagger}(\theta_{1,b})\mathsf{N}(D\Phi_{c,b}) & J_{e2}(\theta_{b+1,n}) \end{bmatrix}$

2.6 Manipulability Indexes

A variety of manipulability indexes have been proposed for evaluation of the performance of manipulators since the first index in [81]. This section discusses some of these manipulability indexes found in literature.

- **Manipulability** [81]

  The manipulability is a numeric index that represents the manipulator distance to singular configurations, thus, maximizing this index means that the manipulator move away from singularities. For a given Jacobian matrix $J(\theta)$ the manipulability measure $w(\theta) \in \mathbb{R}$ is:

  $$w(\theta) = \sqrt{det(J(\theta)J^T(\theta))}. \tag{2.49}$$

- **Manipulability Index Squared** [75]

  The manipulability index squared is a simplified expression that is used because is a convex function and have a less complicated analytical expression:

  $$w^2(\theta) = det(J(\theta)J^T(\theta)). \tag{2.50}$$

- **Velocity Manipulability** [69]

  The velocity manipulability ellipsoid represents the behavior of a manipulator to arbitrarily change end-effector position an orientation. The end-effector maximum velocity at a direction is directly proportional to the length of the ellipsoid axis in this direction. If the ellipsoid is a sphere the maximum end-effector velocity is isotropic (same value when measured in different directions).

  The directions of the principal axes of the ellipsoid are determined by the eigenvectors of the matrix $J(\theta)J^T(\theta)$, while the dimensions of these axes are determined by the eigenvalues of the same matrix.

- **Force Manipulability** [69]

  The force manipulability ellipsoid characterizes the end-effector forces that can be generated with a given set of joint torques being a manipulator at given posture. The end-effector maximum force at a direction is directly proportional to the length of the ellipsoid axis in this direction. The maximum force isotropy is attainable when the ellipsoid is a sphere.

  The force ellipsoid is a dual of the velocity ellipsoid, based on the duality between differential kinematics and statics. So the directions of the principal axes of the ellipsoid are determined by the eigenvectors of the matrix $(J(\theta)J^T(\theta))^{-1}$, while the dimensions of these axes are determined by the eigenvalues of the same matrix.

- **Relative Manipulability** [40]

  The relative manipulability is the manipulability measure independent of the number the degrees of freedom of the manipulator as well as links lengths. It has the following expression:

$$w_n(\theta) = \sqrt[n]{det(J(\theta)J^T(\theta))}/l_w^2, \tag{2.51}$$

  where $l_w = \sum_{i=0}^{n} l_{wi} \in \mathbb{R}$ is the total length of the manipulator, $l_{wi} = \sqrt{a_{wi}^2 + d_{wi}^2} \in \mathbb{R}$ is the total length of the i-th link, $a_{wi} \in \mathbb{R}$ is the i-th link length defined in Denavit-Hartenberg convention and $d_{wi} \in \mathbb{R}$ is the i-th joint offset defined in Denavit-Hartenberg convention. For a explanation of Denavit-Hartenberg convention see [34, 69, 72].

- **Manipulability Polytope** [46]

  The manipulability polytope gives a representation of velocity bounds of the manipulator it transforms the admissible range of velocities from joint space to a polytope in task space. Polytopes are less frequently used than ellipsoids due to the additional computational cost.

  For a manipulator with n joints a joint space polytope that encapsulates all possible joints velocities from the following vertex representation:

$$\theta_v = \begin{bmatrix} \dot{\theta}_1^v \\ \dot{\theta}_2^v \\ \vdots \\ \dot{\theta}_{2n}^v \end{bmatrix} = \begin{bmatrix} \dot{\theta}_1^- & \dot{\theta}_2^- & \cdots & \dot{\theta}_n^- \\ \dot{\theta}_1^- & \dot{\theta}_2^- & \cdots & \dot{\theta}_n^+ \\ \vdots & \dots & \ddots & \vdots \\ \dot{\theta}_1^+ & \dot{\theta}_2^+ & \cdots & \dot{\theta}_n^+ \end{bmatrix}, \tag{2.52}$$

  where $\theta_v \in \mathbb{R}^{2n \times n}$, $\theta_i^v \in \mathbb{R}^n$ and $\theta_i^- \in \mathbb{R}$ and $\theta_i^+ \in \mathbb{R}$ are the minimum and maximum velocity of the i-th joint. Using the manipulator Jacobian is possible to transform the joint space polytope in a task space polytope:

$$v_i = J_e(\theta)\dot{\theta}_i^{vT}, \tag{2.53}$$

$$V_{wv} = \begin{bmatrix} v_{w1} & v_{w2} & \cdots & v_{w2n} \end{bmatrix}^T, \tag{2.54}$$

  where $v_{wi} \in \mathbb{R}^n$ and $V_{wv} \in \mathbb{R}^{2n \times 6}$. The manipulability polytope is obtained by calculating the enclosed volume of V_v.

- **Avoidance Manipulability** [83]

The avoidance manipulability represents the shape-changeability (avoidance ability) of each intermediate link when a prior task is stated for the end-effector of a redundant manipulator. This manipulability is defined when some part of the manipulator (regardless of the end-effector) has to execute a sub-task as example the avoidance of an obstacle. The desired velocity pose of the manipulator $\dot{p}_d$ is defined using the desired joint angle vector derivative $\dot{\theta}_d$ by the relation:

$$\dot{p}_d = J_{Ae}(\theta)\dot{\theta}_d, \tag{2.55}$$

which is expanded in:

$$\dot{\theta}_d = J^{\dagger}_{Ae}(\theta)\dot{p}_d + (I_n - J^{\dagger}_{Ae}(\theta)J_{Ae}(\theta))l_{Aw}, \tag{2.56}$$

where $l_{Aw} \in \mathbb{R}^n$ is an arbitrary vector for the avoidance sub-task executed executed by the manipulator with the redundant degree of freedom. The relation for the i-th link desired velocity $\dot{p}_{id} \in \mathbb{R}^m$ is defined by:

$$\dot{p}_{id} = J_{Ai}(\theta)\dot{\theta}_d, \tag{2.57}$$

where $J_{Ai}(\theta) \in \mathbb{R}^{m\times n}$ is the horizontal concatenation of partial analytical Jacobian until the i-th link with an all zero matrix $0_{m\times n-i}$. By substituting 2.56 into 2.57 the following relation is defined:

$$\dot{p}_{id} = J_{Ai}(\theta)J^{\dagger}_{Ae}(\theta)\dot{p}_d + J_{Ai}(\theta)(I_n - J^{\dagger}_{Ae}(\theta)J_{Ae}(\theta))l_{Aw}. \tag{2.58}$$

Two variables are defined:

$$\Delta\dot{p}_{id} = \dot{p}_{id} - J_{Ai}(\theta)J^{\dagger}_{Ae}(\theta)\dot{p}_d, \tag{2.59}$$

$$M_{wi} = J_{Ai}(\theta)(I_n - J^{\dagger}_{Ae}(\theta)J_{Ae}(\theta)), \tag{2.60}$$

where $\Delta\dot{p}_{id} \in \mathbb{R}^m$ is the avoidance velocity and $M_{wi} \in \mathbb{R}^{m\times n}$ is the avoidance matrix of the i-th link. The shape of the avoidance manipulability ellipsoid is given for the equation:

$$\Delta\dot{p}^T_{id}{M^{\dagger}_{wi}}^T M^{\dagger}_{wi}\Delta\dot{p}_{id} \leq 1. \tag{2.61}$$

The $rank(M_{wi})$ determines the possible avoidance dimension of the i-th link while the singular values of M_{wi} indicates the avoidance ability of the same i-th link.

- **Extended Manipulability** [77]

The extended manipulability consider constraints that limit the manipulator maneuverability in the task space incorporating penalization terms. Any constraint can be incorporated, the discussion relies only on joint boundaries. A joint limit derivative potential function for the i-th joint is defined by:

$$h_{\theta i} = \frac{(\theta_i - \theta_i^-)^2(2\theta_i - \theta_i^+ - \theta_i^-)}{4(\theta_i^+ - \theta_i)^2(\theta_i - \theta_i^-)^2}, \tag{2.62}$$

where θ_i^- and θ_i^+ are the lower and the upper limit of the i-th joint, respectively. The joint boundaries terms are defined as:

$$p_{\theta i}^- = \left\{ \begin{array}{cc} 1, & |\theta_i - \theta_i^-| > |\theta_i^+ - \theta_i| \\ \frac{1}{\sqrt{1+|h_{\theta i}|}}, & \text{otherwise} \end{array} \right. , \tag{2.63}$$

$$p_{\theta i}^+ = \left\{ \begin{array}{cc} \frac{1}{\sqrt{1+|h_{\theta i}|}}, & |\theta_i - \theta_i^-| > |\theta_i^+ - \theta_i| \\ 1, & \text{otherwise} \end{array} \right. , \tag{2.64}$$

where $p_{\theta i}^-$ is applied when $\dot{\theta}_i < 0$ and $p_{\theta i}^+$ when $\dot{\theta}_i > 0$, $\dot{\theta}_i \in \mathbb{R}$ is the joint velocity of the i-th joint.

A augmented Jacobian J_{em} if formed modifying each element of the manipulator Jacobian:

$$J_{emi,j}(\theta) = \left\{ \begin{array}{cc} p_{\theta i}^- J_{i,j}(\theta), & \dot{\theta}_i < 0 \\ p_{\theta i}^+ J_{i,j}(\theta), & \dot{\theta}_i > 0 \end{array} \right. , \tag{2.65}$$

where $J_{i,j}(\theta)$ is the element of row i and column j of $J(\theta)$. The extended manipulability is defined as:

$$w_{em}(\theta) = \sqrt{det(J_{em}(\theta)J_{em}^T(\theta))}. \tag{2.66}$$

- **Null Space Manipulability** [66]

  The null space manipulability is a local measure of the amount of dexterity that is retained when a manipulator has one or more joints failures. The value of a null space manipulability index ranges from zero to one. A zero value indicates a local loss of full end-effector control while a value of one indicates that the joints only produce self motion. Let $J_{wrm}(\theta)$ be the manipulator Jacobian after the columns of the corresponding failed joints being removed. The null space manipulability is defined by:

$$w_{rm}(\theta) = \sqrt{det(\mathsf{N}(J_{wrm}(\theta))\mathsf{N}(J_{wrm}^T(\theta)))}. \tag{2.67}$$

- **Manipulability of Constrained Manipulators, first index** [26]

In order to analyze the manipulability of a constrained serial manipulator [26] proposes the study of two Jacobian matrices, the geometric Jacobian until the joint before the constraint $J_b(\theta_{1,b})$ and the constrained Jacobian $J_r(\theta_{b+1,n})$.

The manipulability of related to $J_b(\theta_{1,b})$ indicates the ability of the constrained manipulator generating motions in F_c in order to track the desired trajectory of the end-effector.

$$w_b(\theta_{1,b}) = \sqrt{det(J_b(\theta_{1,b}){J_b}^T(\theta_{1,b}))}. \tag{2.68}$$

- **Manipulability of Constrained Manipulators, second index** [26]:

  For the constrained Jacobian matrix $J_r(\theta_{b+1,n})$, which can only depend on the constraint type and kinematics of the joints after the constraint, the manipulability indicates the possibility of generating the desired trajectory in the end-effector associated with the use of u_f and $\dot{\theta}_{b+1,n}$:

$$w_r(\theta_{b+1,n}) = \sqrt{det(J_r(\theta_{b+1,n}){J_r}^T(\theta_{b+1,n}))}. \tag{2.69}$$

- **Other Indexes**

  Many other manipulability indexes are defined in the literature, as these indexes are out of scope of this thesis they are only pointed. The indexes related to the dynamic features of robots are: dynamic manipulability ellipsoid [67]; energy manipulability ellipsoid [53]; zero moment point manipulability ellipsoid [55]; dynamic reconfiguration manipulability ellipsoid [23]; dynamic manipulability of the center of mass [4]. For parallel robots there is the power manipulability [47]. The indexes related to teleoperation are: teleoperation manipulability index [76]; infinite manipulability [79]. In [39] are defined the following indexes for continuum robots, velocity manipulability, compliance manipulability and unified force-velocity manipulability. For robots hands the indexes are: joint torque-velocity pair set manipulability [80]; force directional manipulability [60].

Chapter 3

Methods for Trajectory Tracking

In this chapter the objective is to present different methods to solve the following control problem: the end-effector of a serial manipulator tracks a desired trajectory while the holonomic constraints in the kinematic chain of the manipulator are satisfied and the manipulability is maximized.

Three different methods are discussed: in Section 3.1 the kinematic control is presented while in Section 3.2 the quadratic programming is introduced. Lastly, in Section 3.3 the sequential quadratic programming is designed.

The Section 3.4 summarizes the features and presents a comparison among the three methods used for trajectory tracking.

3.1 Kinematic Control

In this section an analytical approach for kinematic control in Cartesian space is presented. This scheme have been used in [16, 17, 62, 63] in the context of robotic-assisted minimally invasive surgery.

Utilizing the constrained Jacobian $J_r(\theta_{k+1,n})$ in (2.48) is possible to obtain the end-effector velocity:

$$V_e = J_r(\theta_{b+1,n}) \begin{bmatrix} u_f \\ \dot{\theta}_{n+1,b} \end{bmatrix}. \tag{3.1}$$

Considering the transition function in (3.1) the Figure 3.1 shows a block diagram for a kinematic control loop with a scleronomic constraint in task space.

Still, in Figure 3.1 applying the forward kinematics results in p_e. The pose error $e_p \in \mathbb{R}^6$ is, same equation of (2.32):

$$e_p = p_d - p_e. \tag{3.2}$$

For a control signal $u_p = K_p e_p + \dot{p}_d$, where $K_p \in \mathbb{R}^{6\times 6}$ is the gain matrix, $\dot{p}_d \in \mathbb{R}^6$ is the time derivative of p_d and $u_p \in \mathbb{R}^6$ is a proportional plus feed forward controller.

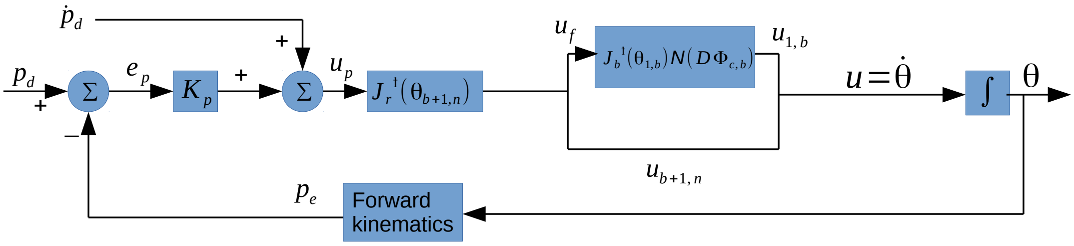


Figure 3.1: Kinematic control loop with scleronomic constraint. For a rheonomic constrain $u_{1,b} = J_b{}^{\dagger}(\theta_{1,b})(\mathsf{N}(D\Phi_{c,b})u_f + (D\Phi_{c,b})^{\dagger}v_d(t))$.

Using (3.1) and u_p, the constrained velocity vector $u_c \in \mathbb{R}^{n-m}$ is given by:

$$u_c = \begin{bmatrix} u_f \\ u_{b+1,n} \end{bmatrix} = J_r{}^{\dagger}(\theta_{b+1,n})u_p, \tag{3.3}$$

where $u_f \in \mathbb{R}^{b-m}$ is equal to the first $b-m$ elements of u_c and $u_{b_1,n} \in \mathbb{R}^{b-m}$ is equal to the last $n-b$ elements of u_c.

For a scleronomic constraint, (2.43) i.e $D\Phi_{c,b}J_b(\theta_{1,b})\dot{\theta}_{1,b} = 0$ holds and $u_{1,b}$ is obtained using $u_f \in \mathbb{R}^{b-m}$ from (3.3). So, the result is:

$$u_{1,b} = J_b{}^{\dagger}(\theta_{1,b})\mathsf{N}(D\Phi_{c,b})u_f. \tag{3.4}$$

For a rheonomic constraint, (2.43) is rewritten as:

$$D\Phi_{c,b}J_b(\theta_{1,b})\dot{\theta}_{1,b} = v_d(t). \tag{3.5}$$

So, $\dot{\theta}_{1,b}$ is defined as:

$$u_{1,b} = J_b{}^{\dagger}(\theta_{1,b})(\mathsf{N}(D\Phi_{c,b})u_f + (D\Phi_{c,b})^{\dagger}v_d(t)). \tag{3.6}$$

The velocity control command, $u \in \mathbb{R}^n$, sent to the manipulator is the vertical concatenation of $u_{1,b}$ from (3.4) or (3.6) and $u_{b+1,n}$ from (3.3):

$$u = \begin{bmatrix} u_{1,b} \\ u_{b+1,n} \end{bmatrix}. \tag{3.7}$$

Considering a scleronomic constraint, u can be rewritten in a matrix multiplication form using the terms of right side of equalities (3.3) and (3.4):

$$u = \begin{bmatrix} {J_b}^{\dagger}(\theta_{1,b})\mathsf{N}(D\Phi_{c,b}) & 0_{b,n-b} \\ 0_{n-b,b-m} & I_{n-b} \end{bmatrix} {J_r}^{\dagger}(\theta_{b+1,n})u_p \tag{3.8}$$

For the velocity command u (3.1) is rewritten as:

$$V_e = J_r(\theta_{b+1,n}) \begin{bmatrix} u_f \\ u_{n+1,b} \end{bmatrix}. \tag{3.9}$$

In (3.9) V_e also can be rewritten in a matrix multiplication form because $u_f = (J_b^{\dagger}(\theta_{1,b})\mathsf{N}(D\Phi_{c,b}))^{\dagger}u_{1,b}$, so:

$$V_e = J_r(\theta_{b+1,n}) \begin{bmatrix} {J_b}^{\dagger}(\theta_{1,b})\mathsf{N}(D\Phi_{c,b}) & 0_{b,n-b} \\ 0_{n-b,b-m} & I_{n-b} \end{bmatrix}^{\dagger} u. \tag{3.10}$$

Now applying the same methodology from Section 2.3.1 in order to obtain the position error dynamics derivative (3.2), substitutes $\dot{p}_d$ with (2.33), i.e $u_p = K_p e_p + \dot{p}_d$ and considering that $\dot{\theta} = u$, substituting (3.8) in (3.10) implies $V_e = u_p$:

$$\dot{e}_p = \dot{p}_d - V_e = u_p - K_p e_p - V_e = -K_p e_p \tag{3.11}$$

where with a positive definite K_p matrix implies that $\lim_{t\to\infty} e_p(t) = 0$.

The control strategy in this Section, that is applied in [63], does not address the manipulability indexes of Section 2.6, it only strives to follow a trajectory with the holonomic constraints satisfied. So, a modification is proposed, which consists in expand the null space of $J_r(\theta_{b+1,n})$ and $J_b(\theta_{1,k})$ rewriting (3.3), (3.4), (3.6) respectively as:

$$\begin{aligned} \begin{bmatrix} u_f \\ u_{b+1,n} \end{bmatrix} &= {J_r}^{\dagger}(\theta_{b+1,n})u_p + \\ &(I_{n-b} - {J_r}^{\dagger}(\theta_{b+1,n})J_r(\theta_{b+1,n}))\mu_r, \end{aligned} \tag{3.12}$$

$$\begin{aligned} u_{1,b} &= {J_b}^{\dagger}(\theta_{1,b})\mathsf{N}(D\Phi_{c,b})u_f + \\ &(I_b - {J_b}^{\dagger}(\theta_{1,b})J_b(\theta_{1,b}))\mu_b, \end{aligned} \tag{3.13}$$

$$\begin{aligned} u_{1,b} = {J_b}^{\dagger}(\theta_{1,b})(\mathsf{N}(D\Phi_{c,b})u_f + (D\Phi_{c,b})^{\dagger}v_d(t)) + \\ (I_b - {J_b}^{\dagger}(\theta_{1,b})J_b(\theta_{1,b}))\mu_b, \end{aligned} \tag{3.14}$$

where μ_b and μ_b are additional degrees of freedom that are utilized for maximize a function, in this case the manipulability indexes defined in (2.68) and (2.69), respectively given by:

$$\mu_b = k_b \left(\frac{\partial w_b(\theta_{1,b})}{\partial \theta} \right)^T, \tag{3.15}$$

$$\mu_r = k_r \left(\frac{\partial w_r(\theta_{b+1,n})}{\partial \theta} \right)^T, \tag{3.16}$$

where $k_b \in \mathbb{R}$ and $k_r \in \mathbb{R}$ define the weight of (3.15) and (3.16), respectively. A kinematic control algorithm for trajectory tracking is defined by Algorithm 2.

Algorithm 2 Kinematic control algorithm.

Define desired trajectory $p_d(t)$
Define constraint $v_d(t)$
Define sampling interval T
repeat
  $e_p \leftarrow p_d - p_e$
  $u_p \leftarrow K_p e_p + \dot{p}_d$
  $u_f \leftarrow$ first $b - m$ lines of (3.12)
  $u_{b+1,n} \leftarrow$ last $n - b$ lines of (3.12)
  if $v_d(t) = 0$ **then**
    $u_{1,b} \leftarrow$ first b lines of (3.13)
  else if $v_d(t) \neq 0$ **then**
    $u_{1,b} \leftarrow$ first b lines of (3.14)
  end if
  $u \leftarrow \begin{bmatrix} u_{1,b} \\ u_{b+1,n} \end{bmatrix}$
until trajectory ends

3.2 Quadratic Programming

In this section the trajectory tracking problem for constrained redundant manipulators is addressed by the QP method. The definition of quadratic optimization is in Subsection 3.2.1 while Subsection 3.2.2 describes how to formulate the tracking problem as a quadratic problem.

3.2.1 Quadratic Optimization

Engineering optimization is a technique that utilizes a method (generally an algorithm or heuristic) to find a way of designing and operating a system. The optimization is done using a combination of the so called decision variables, finding a solution under certain objectives that also satisfies the design and operation constraints of the system.

A QP is an optimization problem with a quadratic objective function and linear constraints, which is defined as:

$$\min_{u} \ \frac{1}{2}u^T C u + c^T u \in \mathbb{R}, \text{subject to:} \tag{3.17}$$

$$\mathcal{F} = \begin{cases} {B_i}^T u = s_i, & i \in \mathcal{E}; \\ {B_i}^T u < s_i, & i \in \mathcal{I}; \\ \quad u \in \mathcal{U}, \end{cases} \tag{3.18}$$

where $u \in \mathbb{R}^n$ is the decision variable vector, $C \in \mathbb{R}^{n \times n}$ is a symmetric matrix of objective function, $c \in \mathbb{R}^n$ is a vector of objective function, $\mathcal{F}$ is the constraint set of the optimization problem, $B_i \in \mathbb{R}^n$ and $s_i \in \mathbb{R}$ define the i-th constraint, $\mathcal{E}$ is the set of equality constraints, $\mathcal{I}$ is the set of inequality constraints and $\mathcal{U}$ is the set of u.

It can be noted in (3.18) that equality and inequality constraints from the optimization problem are linear. In this way, all equations modeling the trajectory tracking problem should be linear relating to decision variables. The same concept is applied for the vector in the objective function.

3.2.2 Trajectory Tracking with QP

Regarding the trajectory tracking problem in manipulators, the end-effector follows the desired pose p_d, this way the joint trajectory have to be determined in real time. A manner to relate the desired pose and the joint trajectory in a linear way is to take the time derivatives of p_d and θ using the analytical Jacobian,

$$J_{Ae}(\theta)\dot{\theta} = \dot{p}_d, \tag{3.19}$$

which is used in the repetitive motion planning scheme in [21, 84, 85].

A modification of (3.19) by adding proportional term information is defined by:

$$J_{Ae}(\theta)\dot{\theta} = \dot{p}_d + k_1(p_d - p_e) \tag{3.20}$$

where $k_1 \in \mathbb{R}^+$. Considering the error $e_p = p_d - p_e$ in (3.20) leads to:

$$J_{Ae}(\theta)\dot{\theta} = \dot{p}_d + k_1 e_p. \tag{3.21}$$

Holonomic constraints in the manipulator kinematic chain are linear equalities regarding $\dot{\theta}$, $D\Phi_{c,b}J_b(\theta_{1,b})\dot{\theta}_{1,b} = v_d(t)$. This way can, they be easily integrated in the quadratic programming formulation as long the decision variable vector u, a joint velocity command for the robot, is equal to $\dot{\theta}$.

To incorporate the manipulability index $w(\theta)$ in the quadratic programming formulation, [21] proposes the following second-order approximation:

$$w(\theta) \approx \nabla w^T(\theta)u + \frac{1}{2}u^T H_w(\theta)u, \tag{3.22}$$

where $\nabla w(\theta)$ is the gradient of $w(\theta)$ and $H_w(\theta)$ is the Hessian of $w(\theta)$.

The gradient $\nabla f(u) \in \mathbb{R}^n$ of a differentiable function in $u \in \mathcal{U}$ is given by:

$$\nabla f(u) = \begin{pmatrix} \frac{\partial f}{\partial u_1} & \frac{\partial f}{\partial u_2} & \cdots & \frac{\partial f}{\partial u_{n-1}} & \frac{\partial f}{\partial u_n} \end{pmatrix}, \tag{3.23}$$

where u_i is the i-th variable.

The Hessian $H(u) \in \mathbb{R}^{n \times n}$ of a second order differentiable function in $u \in \mathcal{U}$ is:

$$H(u) = \begin{pmatrix} \frac{\partial^2 f}{\partial u_1^2} & \frac{\partial^2 f}{\partial u_1 \partial u_2} & \cdots & \frac{\partial^2 f}{\partial u_1 \partial u_{n-1}} & \frac{\partial^2 f}{\partial u_1 \partial u_n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial^2 f}{\partial u_n \partial u_1} & \frac{\partial^2 f}{\partial u_n \partial u_2} & \cdots & \frac{\partial^2 f}{\partial u_n \partial u_{n-1}} & \frac{\partial^2 f}{\partial u_n^2} \end{pmatrix}. \tag{3.24}$$

The manipulability may have a very complex analytical expression. Thus, finding $\nabla w_i(\theta)$ and $H_{wi,j}(\theta)$by the analytical derivative may be impractical. So, [21] proposes the following numerical approximations:

$$\nabla w_i(\theta) = \frac{\partial w}{\partial \theta_i} \approx \frac{w(\theta + \delta\theta_i E_i) - w(\theta - \delta\theta_i E_i)}{2\delta\theta_i}, \tag{3.25}$$

$$H_{wi,j}(\theta) = \frac{\partial^2 w}{\partial \theta_i \partial \theta_j} \approx \frac{\nabla w_j(\theta{+}\delta\theta_i E_i){-}\nabla w_j(\theta{-}\delta\theta_i E_i)}{2\delta\theta_i}, \tag{3.26}$$

where $\delta \in \mathbb{R}$ is a constant and $E_i \in \mathbb{R}^n$ is a null vector except for the i-th element having the value 1.

Now a QP formulation for the trajectory tracking problem considering redundant manipulators that satisfy holonomic constraints in a point of its chain and maximize manipulability indexes is defined by:

$$\begin{aligned} \min_u \; & -\frac{1}{2}u^T(\alpha_b H_{wb}(\theta_{1,b}) + \alpha_r H_{wr}(\theta_{b+1,n}))u \\ & -(\alpha_b \nabla {w_b}^T(\theta_{1,b}) + \alpha_r \nabla {w_r}^T(\theta_{b+1,n}))u \;\in \mathbb{R}, \\ & \qquad\qquad\qquad\qquad\qquad\qquad \text{subject to:} \end{aligned} \tag{3.27}$$

$$J_{Ae}(\theta)u = \dot{p}_d(t) + k_1 e_p; \tag{3.28a}$$

$$D\Phi_{c,b} J_b(\theta_{1,b}) u_{1,b} = v_d(t); \tag{3.28b}$$

$$\theta^- \leq \theta \leq \theta^+; \tag{3.28c}$$

$$\dot{\theta}^- \leq u \leq \dot{\theta}^+, \tag{3.28d}$$

where $H_{wb}(\theta_{1,b})$ and $\nabla w_b(\theta_{1,b})$ are respectively the Hessian and the gradient of $w_b(\theta_{1,b})$, $H_{wr}(\theta_{b+1,n})$ and $\nabla w_r(\theta_{b+1,n})$ are respectively the Hessian and the gradient of $w_r(\theta_{b+1,n})$ while $\alpha_b \in \mathbb{R}$ and $\alpha_r \in \mathbb{R}$ are weights for the manipulability indexes. $\theta^+ \in \mathbb{R}^n$ and $\theta^- \in \mathbb{R}^n$ denote respectively the upper and lower joint angle limits while $\dot{\theta}^+$ and $\dot{\theta}^-$ denote respectively the upper and lower joint velocity limits.

In (3.27) a manipulability is maximized searching through the negative of Hessian and gradient. The first constraint in (3.28a) is responsible for the trajectory tracking. The second constraints in (3.28b) is the holonomic constraint: if $v_d(t)$ is a constant it is a scleronomic constraint; otherwise it is a rheonomic constraint. The last two constraints (3.28c) and (3.28d), the inequalities, are the manipulator physical limits in terms of joint angles and joint velocities, respectively. The QP trajectory tracking is defined by Algorithm 3.

Algorithm 3 QP trajectory tracking algorithm.

Define desired trajectory $p_d(t)$
Define constraint $v_d(t)$
Define sampling interval T
repeat
  $t_{actual} = t$
  $u \leftarrow$ solution of problem in 3.27 and 3.28
  wait until $t_{actual} > t + T$
until trajectory ends

From using QP in order to track a desired trajectory for constrained redundant manipulators the following theorem is established:

Theorem 1. *Considering a redundant holonomic robot system, i.e. a redundant manipulator, where the dynamics effects can be neglected. Assuming joint velocity commands are sent at a fixed rate (a sampling period) for the redundant manipulator and these commands ensures that the following constraints are satisfied at each* *sampling period:*

$$J_{Ae}(\theta)u = \dot{p}_d(t) + k_1 e_p; \tag{3.29a}$$

$$D\Phi_{c,b} J_b(\theta_{1,b}) u_{1,b} = v_d(t); \tag{3.29b}$$

$$\theta^- \leq \theta \leq \theta^+; \tag{3.29c}$$

$$\dot{\theta}^- \leq u \leq \dot{\theta}^+, \tag{3.29d}$$

and also minimizes a convex objective function. This can be stated as a QP problem where the decision variables are equal to joint velocity commands, the constraints are related to trajectory tracking in (3.29a), velocity in frame satisfying a holonomic constraint in (3.29b), manipulator physical limits in joint angles in (3.29c) and manipulator physical limits in joint velocities in (3.29d), also the objective function is the negative of a manipulability index, which can be approximated by a quadratic function. Then, the redundant manipulator will track the desired trajectory in its workspace and satisfy the holonomic constraint assuming it has a high position accuracy, the initial end-effector position coincides with the initial trajectory and the velocity ellipsoids defined by the manipulability indexes $w_b(\theta_1, b)$ *and* $w_m(\theta_{b+1,n})$ *are non vanishing in any direction of the task space.*

Proof. See Appendix A.2. □

3.3 Sequential Quadratic Programming

The objective of this section is describe how the tracking problem can be achieved in constrained redundant manipulators using the SQP method. It starts with the definition of constrained nonlinear optimization and the description of the nonlinear optimization methods in Subsection 3.3.1 while Subsection 3.3.2 discuss the motivation for using SQP in the trajectory tracking problem. 3.3.3 discuss the sequential least squares quadratic programming (SLSQP) , one of many variants implementation of SQP. In Subsection 3.3.4 a brief discussion of multi-objective optimization is done towards the weighted-sum method. Subsection 3.3.5 describes how to formulate the trajectory tracking problem as a constrained nonlinear optimization problem.

3.3.1 Constrained Nonlinear Optimization

The definition of constrained nonlinear optimization problem [28] is - A problem that involves minimization of a nonlinear function subject to constraints (nonlinear or linear) on a finite set of continuous variables. The general formulation of the problem is:

$$\min_u f(u) \in \mathbb{R}, \text{ subject to:} \tag{3.30}$$

$$\mathcal{F} = \begin{cases} g_i(u) = 0, & i \in \mathcal{E}; \\ g_i(u) \leq 0, & i \in \mathcal{I}; \\ u \in \mathcal{U}, \end{cases} \tag{3.31}$$

where $u \in \mathbb{R}^n$ is the decision variable vector, $n \in \mathbb{N}$ is the number of decision variables, $f(u)$ $\mathbb{R}^n \mapsto \mathbb{R}$ is a nonlinear objective function, $\mathcal{F}$ is the constraint set of the optimization problem, $g_i(u)$ $\mathbb{R}^n \mapsto \mathbb{R}$ is the i-th constraint (this constraint can be either linear or nonlinear), $\mathcal{E}$ is the finite set of equality constraints, $\mathcal{I}$ is the finite set of inequality constraints and $\mathcal{U}$ is the set of u.

The Lagrangian $\mathcal{L}(u, \lambda)$ of the constrained nonlinear optimization problem defined in (3.30) and (3.31) is given by:

$$\mathcal{L}(u, \lambda) = f(u) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i g_i(u). \tag{3.32}$$

where $\lambda_i \in \mathbb{R}$ is the i-th Lagrange multipliers of $g(u)$, $\lambda \in \mathbb{R}^{|\mathcal{E}|+|\mathcal{I}|}$ is the Lagrange multipliers.

There are many methods for solving nonlinear constrained optimization problems, according to [5] these techniques and some of their main features are summarized into the text, for a deeper discussion of each method see [5, 6, 56].

a) Reduced-Gradient Method

In the reduced-gradient methods the decision variables are separated into a set of dependent variables (the variables are expressed in terms of other variables) and a set of independent variables. The reduced gradient is computed to find the minimum in the search direction until convergence is achieved. An algorithm for the reduced-gradient method considering the QP formulation of (3.17,3.18) is defined by Algorithm 4, where $range(\cdot)$ is the column space of a matrix.

Each iteration makes a horizontal move in the subspace to satisfy the constraints. The line search is necessary because $f(x)$ in nonlinear. The quasi-newton update keeps B_{k+1} positive-definite. In order to work with nonlinear constraints the reduced-gradient method needs a restoration step to regain feasibility.

b) Penalty Function Method

In the penalty function method a penalty parameter is associated with the objective function in (3.30) leading to the penalty function:

$$f(u) + \frac{1}{r} \sum_{i=1}^{|\mathcal{E}|+|\mathcal{I}|} g_i(u)^2, \tag{3.33}$$

Algorithm 4 Reduced-gradient methods algorithm.

Define initial feasible solution u_0
$\lambda_0 \leftarrow 0$
$\bar{H} \leftarrow \nabla^2 f(u_0)$
$\bar{Y} \leftarrow range(B)$
$\bar{Z} \leftarrow null(B)$
$k \leftarrow 0$
repeat
  $\bar{c}_k \leftarrow \nabla f(u_k)$
  Determine $\bar{z}$, $\bar{Z}^T H_k \bar{Z}\bar{z} = -\bar{Z}^T \bar{c}_k$
  $\bar{p}_k \leftarrow \bar{Z}\bar{z}$
  Determine λ_{k+1}, $\bar{Y}^T B^T \lambda = \bar{Y}^T \bar{c}_k + \bar{Y}^T \bar{H}_k \bar{p}_k$
  Line search: $u_{k+1} \leftarrow u_k + \bar{s}\bar{p}_k$, $f(u_{k+1}) < f(u_k)$
  $\bar{H}_{k+1} \leftarrow$ quasi-Newton update of $\bar{H}_k$
  $k \leftarrow k+1$
until $\| \bar{Z}^T \bar{c}_k \| < \epsilon$

where $r \in \mathbb{R}$ is a positive penalty parameter. If the solution is infeasible the summation increases proportional to the square of the constraint violations. An algorithm for penalty function method is defined by Algorithm 5.

Algorithm 5 Penalty function method algorithm.

Define initial solution u_0
Define initial penalty parameter r_1
Define constant $\beta < 1$
$k \leftarrow 1$
repeat
  Determine u_k with a iterative method from u_{k-1}
  $r_{k+1} \leftarrow \beta r_k$
  $k \leftarrow k+1$
until $\| g(u_k) \| < \epsilon$

The penalty function is minimized for a decreasing sequence of the penalty parameter as $\beta < 1$. The solution u_k in Algorithm 5 does not satisfy the constraints until convergence of the algorithm is achieved, this convergence can be divided into two parts. First is the convergence of penalty function to minimum which follows the convergence rate of the iterative method used, for example a quasi-Newton technique. Second is the convergence of the iterative solution u_k which is linear because the difference $\| u_k - u_{final} \|$ (u_{final} is the solution that satisfies $\| g(u_k) \| < \epsilon$) is proportional to r_k.

Numerical difficulties can occur when the penalty parameter approaches zero because the second term in the penalty function dominates $f(u)$ nullifying the effect of the original objective function in 3.30. The penalty function method is better than reduced-gradients methods for nonlinear constraints.

c) Augmented Lagrangian Method

To avoid ill-conditioning difficulties of the penalty function method an augmented Lagragian function is formed by including a linear term involving violated constraints:

$$f(u) + \frac{1}{r} \sum_{i=1}^{|\mathcal{E}|+|\mathcal{I}|} \Big(g_i(u) - \frac{r}{2} v_i\Big)^2, \tag{3.34}$$

where v_i is the i-th term of $v \in \mathbb{R}^{|\mathcal{I}|+|\mathcal{E}|}$, the vector of linear terms. An algorithm for augmented Lagrangian method is defined by Algorithm 6.

Algorithm 6 Augmented Lagrangian method algorithm.

Define initial solution u_0
Define initial penalty parameter r_1
Define constant $\beta < 1$
Define initial linear terms vector v_1
$k \leftarrow 1$
repeat
  Determine u_k with a iterative method from u_{k-1}
  $v_{k+1 \leftarrow v_k - 2g(u_k)/r_k}$
  $r_{k+1} \leftarrow \beta r_k$
  $k \leftarrow k + 1$
until $\parallel g(u_k) \parallel < \epsilon$

The augmented Lagrangian function is minimized for a sequence of values from the penalty parameter and the linear terms vector as the vector tends toward the Lagrange multipliers. The method does not have the same numerical difficulties as the penalty function method because there is no need for the penalty parameter approach to zero. Usually it is more efficient than penalty function method.

d) Sequential Quadratic Programming

The sequential quadratic programming (SQP) is an iterative method for the constrained nonlinear optimization defined in (3.30) and (3.31). In general lines, at each major iteration the SQP defines the Hessian of the Lagrangian in (3.32) that is used to generate a QP subproblem whose solution is used to form a search direction. A SQP algorithm adapted from [56] can be defined by Algorithm 7.

The SQP is solved iteratively with a initial solution $u_0 \in \mathbb{R}^n$, the $k+1$-th solution is obtained from the k-th solution:

$$u_{k+1} = u_k + \gamma_k d_k, \tag{3.35}$$

where $d_k \in \mathbb{R}^n$ is the search direction and $\gamma_k \in \mathbb{R}$ is the step length parameter.

Algorithm 7 SQP algorithm.

Initialize u_0
$k \leftarrow 0$
repeat
  Evaluate $\mathcal{L}(u_k, \lambda_k)$
  Evaluate $\nabla f(u_k)$
  Formulate the QP problem defined by (3.36) and (3.37)
  Solve the QP problem to obtain d_k
  Determine γ_k using the merit function in (3.38)
  $u_{k+1} \leftarrow u_k + d_k\gamma_k$
  $k \leftarrow k+1$
until convergence test is satisfied

At each k-th iteration of SQP the search direction is determined by a quadratic programming subproblem define by:

$$\min_{d_k} \frac{1}{2}{d_k}^T H_k(\mathcal{L}(u_k, \lambda_k))d_k + \nabla f^T(u_k)d_k \in \mathbb{R}, \text{subject to:} \tag{3.36}$$

$$\mathcal{F} = \begin{cases} \nabla g_i^T(u_k)d_k + g_i(u_k) = 0; & i \in \mathcal{E} \\ \nabla g_i^T(u_k)d_k + g_i(u_k) \leq 0; & i \in \mathcal{I} \\ d_k \in \mathcal{D}, & \end{cases} \tag{3.37}$$

where $\mathcal{D}$ is set of d_k.

The step length parameter γ_k is determined to produce a sufficient decrease in a defined merit function $\Psi(u)$ in the way:

$$\Psi(u_k + d_k\gamma_k) > \Psi(u_k). \tag{3.38}$$

The SQP is more efficient than penalty function method or augmented Lagrangian method when constraints are nonlinear and it is competitive with reduced-gradients method for linear constraints.

e) Barrier Function Method

The barrier function method is characterized by generating points inside a feasible region, i.e. solutions that satisfy the constraints. The barrier function is defined including a barrier term involving reciprocals of constraints in the objective function of 3.30:

$$f(u) + r\sum_{i=1}^{|\mathcal{E}|+|\mathcal{I}|} \frac{1}{g_i(u)}, \tag{3.39}$$

so, when u is on the border of a feasible region, some $g_i(u)$ is near zero and the barrier term tends to be much greater than $f(u)$. An algorithm for the barrier function method is defined by Algorithm 8.

Algorithm 8 Barrier function method algorithm.

Define initial solution u_0
Define initial penalty parameter r_1
Define constant $\beta < 1$
$k \leftarrow 1$
repeat
  Determine u_k with a iterative method from u_{k-1}
  $r_{k+1} \leftarrow \beta r_k$
  for $i = 1$ **to** $|\mathcal{E}| + |\mathcal{I}|$ **do**
    $\lambda_i = \frac{r_k}{g_i(u_k)^2}$
  end for
  $k \leftarrow k + 1$
until $\lambda_i g_i(u_k) < \epsilon \ \forall i$

The barrier function is minimized for a decreasing sequence of barrier term values. The method works for problems with inequality constraints only. It is usually less efficient than penalty function method but is still useful if the objective function is not evaluated at infeasible solutions.

f) Interior Point Method

The interior point method is related to barrier functions also introducing slack variables to reformulate the inequalities as equalities and hence obtain the solution for the optimization problem. The objective function with the barrier term is defined as:

$$f(u) + r \sum_{i=1}^{|\mathcal{I}|} \log g_i(u), \tag{3.40}$$

in order to prevent the solution leaving the feasible region defined by the inequalities. An algorithm for interior point method is defined by Algorithm 9.

The interior point method uses line searches to enforce convergence and employ matrix factorization to compute steps. In terms of performance can be competitive with SQP methods for nonlinear constraints and with reduced-gradient methods for linear constraints.

3.3.2 Motivation for Using the SQP Method

After a brief summary of the constrained nonlinear optimization methods a question arises: which is the most suited method for the trajectory tracking problem where a redundant manipulator satisfy holonomic constraints and maximize the manipulability index. Also, it is worth emphasizing that the trajectory tracking problem

Algorithm 9 Interior point method algorithm.

Define initial solution u_0
Define initial penalty parameter r_1
Define constant $\beta < 1$
$k \leftarrow 1$
repeat
  Define the Karush–Kuhn–Tucker (KKT) conditions for the nonlinear problem
  Apply Newton method using KKT to obtain d_k
  Line search with $g_i(u)$ to obtain $\gamma_k = \max\{\gamma_k \in (0,1]\}$
  $u_{k+1} \leftarrow u_k + d_k\gamma_k$
  $r_{k+1} \leftarrow \beta r_k$
  $k \leftarrow k+1$
until Convergence test is satisfied

requires commands to be sent to the manipulator in a finite interval, usually short period, of time.

As it can be seen in Section 3.3.5, the trajectory tracking problem described previously as a constrained nonlinear optimization method has inequalities and nonlinear equalities constraints. The nonlinear constraints eliminate the choice of reduced-gradient methods. The presence of equalities constraints put aside the choice of the barrier function method, also considering that the objective function and constraints can be evaluated at infeasible solutions (joints velocities that surpass the physical limits of the manipulator).

Modeling the tracking problem to send commands to the manipulator at a fixed sampling period, usually a fraction of a second in real world applications, requires optimization to have fast convergence. This characteristic eliminates the choice of penalty function and augmented Lagrangian methods in favor of a SQP method and an interior point method (IPM) .

In optimization, test functions are used to evaluate characteristics of algorithms, such as convergence rate, precision, robustness and general performance. There are thousands of test functions in literature, a collection of some wide spreading test functions can be found in [73]. In [24] is presented a few selected test functions from [1, 30], the description of these functions is in Table 3.1.

In [24] the tests are done using an active set SQP implementation and an IPM implementation (primal-dual implementation based in Lagrange multipliers and Newton's method [5]). Both highly constrained (the decisions variables and constraints have the same order of magnitude) and loosely constrained (the decisions variables have a larger order of magnitude of constraints) problems are evaluated, the results are in Table 3.2 and Table 3.3, respectively.

Table 3.2 shows that the SQP is at least 15 times faster than IPM in highly constrained problems while Table 3.3 shows that the IPM is at least 60 times faster than

Table 3.1: Selected test functions.

Test function	Description
MINC44 [52]	Minimize the permanent (definition in [29]) of a doubly stochastic square matrix (definition in [2]) whose trace is zero.
READING8 [48]	A nonlinear optimal control problem considering tidal power generation.
NCVXQP6	A family of non-convex quadratic problems.
MADSSCHJ [49]	A nonlinear minimax problem with equality and inequality constraints and variable dimension.
JIMACK [36]	3-D discretization in finite element method.
OSORIO [13]	Unified framework from techniques in large-scale tabular data protection.
TABLE8	Same problem from OSORIO with less variables and constraints.
OBSTCLBL [19]	Obstacle problem where a rectangle is discretized in many minor rectangles.

Table 3.2: Highly constrained problems from [24].

Name	Nr. variables	Nr. constraints	SQP time(s)	IPM time(s)
MINC44	1113	1033	0.28	7.60
READING8	2002	1000	9.78	251.12
NCVXQP6	10000	7500	3.60	613.38
MADSSCHJ	201	398	0.34	5.51

Table 3.3: Loosely constrained problems from [24].

Name	Nr. variables	Nr. constraints	SQP time(s)	IPM time(s)
JIMACK	3549	0	542.42	8.12
OSORIO	10201	202	303.00	0.78
TABLE8	1271	72	3.80	0.04
OBSTCLBL	10000	1	40.84	0.50

SQP in loosely constrained problems. The tracking problem previously described in this section for a seven degrees of freedom manipulator is a highly constrained problem, so the SQP is the choice. Lastly [24] summarizes some the advantages of the SQP over IPM which are related to the tracking problem:

- Efficient on highly constrained problems: the tracking problem as modeled in Section 3.3.5 has more constraints than decision variables.

- Stays feasible with respect to the linear constraints throughout the optimization: the velocity commands sent to the manipulator will not attempt to bring the joints angles beyond their limits (joint angles limits will be modeled as linear constraints in Section 3.3.5).

- Usually requires less function evaluations: a fast convergence is necessary in the tracking problem.

- Allows warm starting: the solution and the commands sent to manipulator are used to parameterize the method.

The SQP method is already used in robotics related applications, especially in motion planning, here are presented some works. In [51] the SQP is applied to jointly optimize over the parameters in a task planning trajectory in mobile manipulation. In [45] the SQP seeks the solution for an optimization motion planning problem where a manipulator is mounted in a spacecraft. In [42] a mobile manipulator is expected to follow a trajectory, in an event of failure to obtain a feasible trajectory a deviation in the Cartesian space is calculated using the SQP. In [70] the SQP is used to find a human-like trajectory in a robotic arm-hand system. In [59] a comparison between SQP and IPM is done towards trajectory optimization for robot motion planning.

3.3.3 Sequential Least Squares Quadratic Programming

The sequential least squares quadratic programming [41] is one of many variants of a SQP algorithm. It strives for solving the nonlinear optimization problem defined in (3.30) and (3.31) by using a sequence of constrained least squares problems with successive second order approximations of the objective function and first order approximations of the constraints.

The SLSQP follows the general framework defined by Algorithm 7 starting by choosing an initial solution u_0. The next step is enter the loop and evaluate at each iteration $\mathcal{L}(u_k, \lambda_k)$ and $\nabla f(u_k)$ in order to formulate the QP problem defined by (3.36) and (3.37). For computational efficiency it is imperative that the Hessian

$H_k(\mathcal{L}(u_k, \lambda_k))$ in (3.37) is not calculated by the expression in (3.24), but approximated by some algorithm.

The SLSQP uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) iterative algorithm [56] where the Hessian is approximated using gradient evaluations. In this way, at each iteration of the SLSQP algorithm the BFGS algorithm is called in order to compute the approximated Hessian $\tilde{H}_k(u_k) \in \mathbb{R}^{n\times n}$.

The BFGS is solved iteratively from an initial decision variable $\tilde{u}_0 \in \mathbb{R}^n$ and an initial Hessian matrix $\tilde{H}_0(\tilde{u}_0) \in \mathbb{R}^{n\times n}$, then BFGS enters a loop until convergence is obtained. At each k-th iteration of BFGS the search direction $\tilde{d}_k \in \mathbb{R}^n$ is determined by:

$$\tilde{H}_k(\tilde{u}_k)\tilde{d}_k = -\nabla f(\tilde{u}_k), \tag{3.41}$$

then $\tilde{d}_k$ is used to find a step length parameter $\tilde{\gamma}_k \in \mathbb{R}$ by a line search strategy [56]:

$$\tilde{\gamma}_k = \arg\min_{\tilde{\gamma}>0} f(\tilde{u}_k + \tilde{\gamma}\tilde{d}_k), \tag{3.42}$$

and $\tilde{u}_{k+1}$ is given by:

$$\tilde{u}_{k+1} = \tilde{u}_k + \tilde{\gamma}_k\tilde{d}_k. \tag{3.43}$$

The Hessian approximation in the $k+1$-th iteration of BFGS method is updated as:

$$\tilde{H}_{k+1}(\tilde{u}_{k+1}) = \tilde{H}_k(\tilde{u}_k) + \frac{\tilde{r}_k\tilde{r}_k^T}{\tilde{r}_k^T\tilde{s}_k} - \frac{\tilde{H}_k(\tilde{u}_k)\tilde{s}_k\tilde{s}_k^T\tilde{H}_k^T(\tilde{u}_k)}{\tilde{s}_k^T\tilde{H}_k(\tilde{u}_k)\tilde{s}_k}, \tag{3.44}$$

where $\tilde{r}_k \in \mathbb{R}^n$ is defined as

$$\tilde{r}_k = \nabla f(\tilde{u}_{k+1}) - \nabla f(\tilde{u}_k), \tag{3.45}$$

where $\tilde{s}_k \in \mathbb{R}^n$ is defined as

$$\tilde{s}_k = \tilde{u}_{k+1} - \tilde{u}_k. \tag{3.46}$$

The update of (3.44) guarantees the symmetry and positive definiteness of $\tilde{H}_{k+1}(\tilde{u}_{k+1})$. An algorithm for the BFGS method is defined by Algorithm 10.

The next step in the SLSQP is to formulate the QP problem of (3.36) and (3.37) using $\tilde{H}_k(u_k)$ instead $H_k(\mathcal{L}(u_k, \lambda_k))$. In order to find d_k, used to form a new iterate in (3.35), the SLSQP changes the QP formulation of (3.36) and (3.37) into the following linear least squares (LSEI) formulation [7]:

$$\min_{d_k} \frac{1}{2} \| Ad_k - a \| \in \mathbb{R}, \text{subject to:} \tag{3.47}$$

Algorithm 10 Algorithm for the BFGS method.

Initialize $\tilde{u}_0$
Initialize $\tilde{H}_0(\tilde{u}_0)$
$k \leftarrow 0$
repeat
  $\tilde{d}_k \leftarrow -\tilde{H}_k^{-1}(\tilde{u}_k)\nabla f(\tilde{u}_k)$
  $\tilde{\gamma}_k \leftarrow \arg\min f(\tilde{u}_k + \tilde{\gamma}\tilde{d}_k)$ with $\tilde{\gamma} > 0$
  $\tilde{u}_{k+1} \leftarrow \tilde{u}_k + \tilde{\gamma}_k\tilde{d}_k$
  $\tilde{r}_k \leftarrow \nabla f(\tilde{u}_{k+1}) - \nabla f(\tilde{u}_k)$
  $\tilde{s}_k \leftarrow \tilde{u}_{k+1} - \tilde{u}_k$
  $\tilde{H}_{k+1}(\tilde{u}_{k+1}) \leftarrow \tilde{H}_k(\tilde{u}_k) + \frac{\tilde{r}_k\tilde{r}_k^T}{\tilde{r}_k^T\tilde{s}_k} - \frac{\tilde{H}_k(\tilde{u}_k)\tilde{s}_k\tilde{s}_k^T\tilde{H}_k^T(\tilde{u}_k)}{\tilde{s}_k^T\tilde{H}_k(\tilde{u}_k)\tilde{s}_k}$
  $k \leftarrow k+1$
until convergence test is satisfied

$$\mathcal{F} = \begin{cases} M_{eq}^T d_k = m_{eq}; \\ M_{iq}^T d_k = m_{iq}; \\ d_k \in \mathcal{D}, \end{cases} \tag{3.48}$$

where $A \in \mathbb{R}^{n\times n}$ and $a \in \mathbb{R}^n$ can be found respectively by $\tilde{H}(u_k) = A^T A$ and $\nabla f(u_k) = -A^T a$. $M_{eq} \in \mathbb{R}^{n\times|\mathcal{E}|}$ is the equality constraint matrix, $M_{iq} \in \mathbb{R}^{n\times|\mathcal{I}|}$ is the inequality constraint matrix, $m_{eq} \in \mathbb{R}^n$ is the equality constraint vector and $m_{iq} \in \mathbb{R}^n$ is the inequality constraint vector, which are defined by:

$$M_{eq} = \begin{bmatrix} \nabla g_i(u_k) & \cdots & \nabla g_{|\mathcal{E}|}(u_k) \end{bmatrix};\ i \in \mathcal{E}, \tag{3.49}$$

$$M_{iq} = \begin{bmatrix} \nabla g_i(u_k) & \cdots & \nabla g_{|\mathcal{I}|}(u_k) \end{bmatrix};\ i \in \mathcal{I}, \tag{3.50}$$

$$m_{eq} = \begin{bmatrix} g_i(u_k) \\ \cdots \\ g_{|\mathcal{E}|}(u_k) \end{bmatrix};\ i \in \mathcal{E}, \tag{3.51}$$

$$m_{iq} = \begin{bmatrix} g_i(u_k) \\ \cdots \\ g_{|\mathcal{I}|}(u_k) \end{bmatrix};\ i \in \mathcal{I}. \tag{3.52}$$

The QP problem defined by the LSEI formulation in (3.47) and (3.48) is solved through a non-negative least squares (NNLS) algorithm. The NNLS compute only non negative constraints so there are some variables transformation to make it possible. The first is to use the orthogonal basis $M_{ot} \in \mathbb{R}^{n\times n}$ of the nullspace of M_{eq}^T to rewrite d_k by the following relations:

$$d_k = M_{ot}\begin{bmatrix} d_{k1} \\ d_{k2} \end{bmatrix}, \tag{3.53}$$

$$\begin{bmatrix} M_{eq}^T \\ A \\ M_{iq}^T \end{bmatrix} M_{ot} = \begin{bmatrix} \tilde{M}_{eq1} & 0_{|\mathcal{E}|\times n-|\mathcal{E}|} \\ \tilde{A}_1 & \tilde{A}_2 \\ \tilde{M}_{iq1} & \tilde{M}_{iq2} \end{bmatrix}, \tag{3.54}$$

where $d_{keq} \in \mathbb{R}^{|\mathcal{E}|}$, $d_{kiq} \in \mathbb{R}^{n-|\mathcal{E}|}$, $\tilde{M}_{eq1} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$, $\tilde{A}_1 \in \mathbb{R}^{n\times|\mathcal{E}|}$, $\tilde{A}_2 \in \mathbb{R}^{n\times n-|\mathcal{E}|}$, $\tilde{M}_{iq1} \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{E}|}$ and $\tilde{M}_{iq2} \in \mathbb{R}^{|\mathcal{I}|\times n-|\mathcal{E}|}$. So d_{k1} is determined by the following relation:

$$\tilde{M}_{eq1} d_{k1} = m_{eq}. \tag{3.55}$$

In order to obtain d_{k2} the following inequality constrained least squares problem (LSI) is defined ($\mathcal{D}_2$ i the d_{k2} set):

$$\min_{d_{k2}} \parallel \tilde{A}_2 d_{k2} - (a - \tilde{M}_{eq1} d_{k1}) \parallel \in \mathbb{R}, \text{subject to:} \tag{3.56}$$

$$\mathcal{F} = \begin{cases} \tilde{M}_{iq2} d_{k2} \geq m_{iq} - \tilde{M}_{iq1} d_{k1}; \\ \hfill d_{k2} \in \mathcal{D}_2, \end{cases} \tag{3.57}$$

The LSI problem is not solved, instead it will be transformed to the least distance problem (LPD) with a variable change defined as:

$$d_{k3} = R_{LPD} d_{k2} - \tilde{a}, \tag{3.58}$$

where $d_{k3} \in \mathbb{R}^{|n-\mathcal{E}|}$, $R_{LPD} \in \mathbb{R}^{n-|\mathcal{E}|\times n-|\mathcal{E}|}$ is obtained from the following QR factorization:

$$\tilde{A}_2 = Q_{LPD} \begin{bmatrix} R_{LPD} \\ 0_{|\mathcal{E}|\times n-|\mathcal{E}|} \end{bmatrix}, \tag{3.59}$$

$\tilde{a} \in \mathbb{R}^{n\times|\mathcal{E}|}$ is:

$$\tilde{a} = \tilde{Q}_{LPD}^T (a - \tilde{A}_1 d_{k1}), \tag{3.60}$$

and $\tilde{Q}_{LPD} \in \mathbb{R}^{n\times n-|\mathcal{E}|}$ is the first $n - |\mathcal{E}|$ columns of Q_{LPD}. The LPD problem is defined as ($\mathcal{D}_3$ i the d_{k3} set):

$$\min_{d_{k3}} \parallel d_{k3} \parallel \in \mathbb{R}, \text{subject to:} \tag{3.61}$$

$$\mathcal{F} = \begin{cases} \tilde{M}_{iq2} R_{LPD}^{-1} d_{k3} \geq m_{iq} - (\tilde{M}_{iq1} d_{k1} + \tilde{M}_{eq2} R_{LPD}^{-1} \tilde{a}); \\ \hfill d_{k3} \in \mathcal{D}_3, \end{cases} \tag{3.62}$$

The LPD has a dual non-negative least squares problem (NNLS) defined as ($\mathcal{D}_3$ i the d_{k3} set):

$$\min_{d_{k4}} \parallel \tilde{A}_3 d_{k4} - [0_{n,1} \ 1]^T \parallel \in \mathbb{R}, \text{subject to:} \tag{3.63}$$

$$\mathcal{F} = \begin{cases} d_{k4} \geq 0; \\ d_{k4} \in \mathcal{D}_4, \end{cases} \tag{3.64}$$

where $d_{k4} \in \mathbb{R}^{|\mathcal{E}|+|\mathcal{I}|}$ and $\tilde{A}_3$ is defined as:

$$\tilde{A}_3 = Q_{LPD} \begin{bmatrix} \tilde{M}_{iq} & \tilde{M}_{eq} \\ \left[m_{iq} - (\tilde{M}_{iq1} d_{k1} + \tilde{M}_{eq2} R_{LPD}^{-1} \tilde{a})\right]^T & 0_{|\mathcal{E}|\times 1} \end{bmatrix}. \tag{3.65}$$

In order to find the solution the NLLS set some system variables to zero creating the active set. In this way the non-negative constraint of these variables is active. In each iteration the active set is modified by one variable and is ignored leading to a solution of an unconstrained least squares subproblem, until convergence is achieved. Details about the NNLS implementation can be found in [14].

The residue $d_{k5} \in \mathbb{R}^{n+1}$of the NNLS problem is defined by:

$$d_{k5} = \tilde{A}_3 d_{k4} - [0_{n,1}\ 1]^T. \tag{3.66}$$

Each i-th term solution from the the LDP problem can be determined using the i-th element of the NNLS residue:

$$d_{k3_i} = \frac{d_{k5_i}}{d_{k5_{n+1}}},\ i = 1, \ldots, n - |\mathcal{E}|. \tag{3.67}$$

To obtain d_{k2} just use (3.58) with d_{k3} from (3.67), reminding that d_{k1} was defined by (3.55) it is now possible use (3.53) to finally define the solution of LSEI problem, d_k the search direction of SLSQP method. An algorithm for the resolution of LSEI problem is defined by Algorithm 11.

Algorithm 11 Algorithm for the LSEI problem.

Formulate the LSEI problem in (3.47) and (3.48)
Define d_{k1} and d_{k2} from (3.53)
Determine M_{ot} and other matrices from (3.54)
$d_{k1} \leftarrow \tilde{M}_{eq1}^{-1} m_{eq}$
Formulate the LSI problem in (3.56) and (3.57)
QR factorization in (3.59)
Formulate the LPD problem in (3.61) and (3.62)
Formulate the dual NNLS problem in (3.63) and (3.64)
Determine NNLS residue: $d_{k5} \leftarrow \tilde{A}_3 d_{k4} - [0_{n,1}\ 1]^T$
Determine LPD solution: $d_{k3_i} = \frac{d_{k5_i}}{d_{k5_{n+1}}}$
$d_{k2} \leftarrow R_{LPD}^{-1}(d_{k3} + \tilde{a})$
$d_k \leftarrow M_{ot} \begin{bmatrix} d_{k1} \\ d_{k2} \end{bmatrix}$

After finding the search direction, solution of QP problem, the next step of

SLSQP is to determine the step length parameter. An overall value would be $\gamma_k = 1$ but if u_k is far from a local optimum this value will not guarantee the convergence. The following merit function is defined:

$$\Psi(u_k) = f(u_k) + \sum_{i=1}^{|\mathcal{E}|} \varphi_i |g_i(u_k)| + \sum_{i=|\mathcal{E}|+1}^{|\mathcal{E}|+|\mathcal{I}|} \varphi_i |\min(0, g_i(u_k))|, \tag{3.68}$$

where $\min(\cdot,\cdot)$ is the minimum value between two arguments and $\varphi_i \in \mathbb{R}$ is defined by:

$$\varphi_i = \max\left(\frac{1}{2}(\varphi_{i_{k-1}} + |\lambda_i|), |\lambda_i|\right),\ i = 1, \ldots, |\mathcal{E}| + |\mathcal{I}|, \tag{3.69}$$

where $\max(\cdot,\cdot)$ is the maximum value between two arguments, $\varphi_{i_{k-1}}$ is the value for φ_i in the $k-1$-th iteration of SLSQP and λ_i is the Lagrange multiplier of the i-th constraint. An algorithm for the resolution of SLSQP is defined by Algorithm 12.

Algorithm 12 Algorithm for the SLSQP problem.

Initialize u_0
$k \leftarrow 0$
repeat
  $\tilde{H}_k(u_k) \leftarrow$ solution of BFGS algorithm
  Formulate the LSEI problem
  $d_k \leftarrow$ solution for the LSEI algorithm
  Solve the QP problem to obtain d_k
  $\gamma_k \leftarrow$ satisfy: $\Psi(u_k + d_k\gamma_k) > \Psi(u_k)$
  $u_{k+1} \leftarrow u_k + d_k\gamma_k$
  $k \leftarrow k + 1$
until convergence test is satisfied

3.3.4 Multi-Objective Optimization

The general formulation of a constrained nonlinear multi-objective optimization problem is:

$$\min_u f_i(u) \in \mathbb{R},\ i \in \mathcal{M},\ \text{subject to:} \tag{3.70}$$

$$\mathcal{F} = \begin{cases} g_i(u) = 0, & i \in \mathcal{E}; \\ g_i(u) \leq 0, & i \in \mathcal{I}; \\ \quad u \in \mathcal{U}, & \end{cases} \tag{3.71}$$

where $f_i(u)\mathbb{R}^n \mapsto \mathbb{R}$ is the i-th objective function (at least one objective have to be nonlinear) and $\mathcal{M}$ is the finite set of objective functions.

In the multi-objective problem defined by (3.70) and (3.71) $|\mathcal{M}|$ objective functions have to be minimized at the same time, however the functions can be conflicting, that means a minimization of one objective function implies in maximization of another. This problem can have a huge or infinite number of solutions so a method to compare these solutions is required.

Let $u_1 \in \mathcal{U}$ and $u_2 \in \mathcal{U}$ be solutions of the multi-objective problem. u_1 dominates u_2 if $f_i(u_1) \leq f_i(u_2), i \in \mathcal{M}$ and $f_i(u_1) \neq f_i(u_2), i \in \mathcal{M}$, that is, at least in one objective the inequality is strict. This dominance relation is defined by the following notation [18]:

$$u_1 \prec u_2. \tag{3.72}$$

If $u_1 \in \mathcal{U}$ and $u_2 \in \mathcal{U}$ are non dominated among themselves:

$$u_1 \nprec u_2 \text{ and } u_2 \nprec u_1. \tag{3.73}$$

A solution $u^* \in \mathcal{U}$ is globally Pareto-optimal if there is no solution $u \in \mathcal{U}$ that dominates u^*. So the global Pareto-optimal set that contains only globally Pareto-optimal solutions is defined by:

$$\mathcal{P} = \{u^* \in \mathcal{U} \mid \nexists\ u \in \mathcal{U} \mid f(u_2) \nprec f(u_1)\}, \tag{3.74}$$

where the cardinality of $\mathcal{P}$ can be huge or infinity. In real-world engineering problems it is necessary to estimate a finite and representative subset of $\mathcal{P}$.

The weighted-sum is a scalar method to solve multi-objective problems. The original multi-objective problem is transformed is a mono-objective problem using a weighted sum of the original objectives.

The original multi-objective problem in (3.70) and (3.71) is rewritten as:

$$\min_u \sum_{i=1}^{|\mathcal{M}|} \alpha_i f_i(u) \in \mathbb{R},\ i \in \mathcal{M},\ \text{subject to:} \tag{3.75}$$

$$\mathcal{F} = \begin{cases} g_i(u) = 0, & i \in \mathcal{E}; \\ g_i(u) \leq 0, & i \in \mathcal{I}; \\ u \in \mathcal{U}, \end{cases} \tag{3.76}$$

where $\alpha_i \in \mathbb{R}$ is the i-th weight element and $\sum_{i=1}^{|\mathcal{M}|} \alpha_i = 1$.

In the case of only two objective functions (3.75) is rewritten:

$$\min_u\ \alpha_1 f_1(u) + \alpha_2 f_2(u), \tag{3.77}$$

using the fact that $\alpha_1 + \alpha_2 = 1$:

$$\min_u\ (1-\alpha) f_1(u) + \alpha f_2(u), \tag{3.78}$$

where $\alpha \in \mathbb{R}$ is the weight.

In order to generate a set of solutions using this method we have to simply change the weight value in (3.78) subject to (3.76). If the change interval of the weight is small enough, a representative global Pareto-optimal set will be generated for convex objective functions. The main advantage of the weighted-sum method is the ease of programming [3].

3.3.5 Trajectory Tracking with SQP

The pose error is the difference between the desired pose $p_d(t)$ and the actual end-effector pose p_e (time explicit in e_p and θ explicit in p_e):

$$e_p(t) = p_d(t) - p_e(\theta). \tag{3.79}$$

Using the SQP to find a solution $u \in \mathbb{R}^n$, a joint velocity command for the manipulator at a fixed step time $T \in \mathbb{R}$, that aims to bring the pose error in (3.79) to zero in a step time, the predicted pose error $\tilde{e}_p \in \mathbb{R}^6$ is the desired pose after the step time minus the pose after the step time (considering that the solution u is constant at all the step time interval the increment in the joint angle is uT):

$$\tilde{e}_p(t) = p_d(t + T) - p_e(\theta + uT). \tag{3.80}$$

In an optimization problem, a function can be maximized searching through the minimization of the negative direction. So, two functions f_1 and f_2 are defined as the negative of w_b and w_r respectively, and evaluated with the SQP solution:

$$f_1 = -w_b(\theta_{1,b} + u_{1,b}T), \tag{3.81}$$

$$f_2 = -w_r(\theta_{b+1,n} + u_{b+1,n}T). \tag{3.82}$$

For a serial redundant manipulator that satisfy one or more holonomic constraints in a point of this kinematic chain and tracks a trajectory, using (3.81) and (3.82) with a parameter $\alpha \in \mathbb{R}$ where $0 \leq \alpha \leq 1$, the following optimization problem is defined where the decisions variables are the joint velocities commands u_i:

$$\min_{u} \; (1 - \alpha)f_1 + \alpha f_2 \; \in \mathbb{R}, \text{ subject to:} \tag{3.83}$$

$$
\begin{aligned}
&\tilde{e}_p(t) = 0; && (3.84a)\\
&D\Phi_{c,b}J_b(\theta_{1,b} + u_{1,b}T)u_{1,b} = v_d(t+T); && (3.84b)\\
&\theta_i^- \leq \theta_i + u_iT \leq \theta_i^+; && (3.84c)\\
&\dot{\theta}_i^- \leq u_i \leq \dot{\theta}_i^+, && (3.84d)
\end{aligned}
$$

notice that, the decision variables are not explicit in (3.83) and the first equality constraint of (3.84a), the relations are defined in (3.80) to (3.82).

The objective function in (3.83) is minimized at each step of the SLSQP method reflecting in an instantaneous value for w_b and w_r. In order to implement the first equality of (3.84a), that is predicted error is equal to zero, it is used the forward kinematics function evaluated at $\theta + uT$:

$$
\tilde{e}(t) = p_d(t+T) - \mathsf{FK}(\theta + uT) = 0. \tag{3.85}
$$

The second equality in (3.84b) is the holonomic constraint, if $v_d(t)$ is a constant it is a scleronomic constraint, otherwise it is a rheonomic constraint. Notice that, the second equality of (3.84b) is the same expression for the left side of (2.43) but now evaluated with u and T. The last two inequality constraints (3.84c) and (3.84d)are the i-th manipulator physical constraints in terms of joint angle limits and joint velocity limits, respectively.

It is worth mentioning that the u_{k+1} solution stay within the limits $\pm\epsilon$ from the u_k solution, where $\epsilon \in \mathbb{R}$. This is necessary to avoid that u go to the lower and upper velocity limits in consecutive steps of the SQP method. The SQP trajectory tracking is defined by Algorithm 13.

Algorithm 13 SQP trajectory tracking algorithm.

```
Define desired trajectory p_d(t)
Define constraint v_d(t)
Define sampling interval T
repeat
   t_actual = t
   u ← solution of problem in 3.83 and 3.84
   wait until t_actual > t + T
until trajectory ends
```

Remark 1. *The optimization problem of (3.83) and (3.84) can be treated as a unique solution where the α parameter would have to be fixed a priori. This unique solution results in manipulability values that might not be good compared to other attainable values. In fact, there may be a range of α values that make the manipulability indexes* *cooperative and another range where the manipulability indexes are in opposition. Then, only with multiple solutions is possible to verify the correlation between these manipulability indexes and this correlation changes according to the location and type of constraint.*

From using SQP in order to track a desired trajectory for constrained redundant manipulators the following theorem is established:

Theorem 2. *Considering a redundant holonomic robot system, i.e. a redundant manipulator, where the dynamics effects can be neglected. Assuming joint velocity commands are sent at a fixed rate (a sampling period) for the redundant manipulator and these commands ensures that the following constraints are satisfied at each sampling period:*

$$p_d(t+T) - \mathsf{FK}(\theta + uT) = 0; \tag{3.86a}$$

$$D\Phi_{c,b} J_b(\theta_{1,b} + u_{1,b}T)u_{1,b} = v_d(t+T); \tag{3.86b}$$

$$\theta_i^- \le \theta_i + u_i T \le \theta_i^+; \tag{3.86c}$$

$$\dot{\theta}_i^- \le u_i \le \dot{\theta}_i^+, \tag{3.86d}$$

and also minimizes a nonlinear function. This can be stated as a SQP problem where the decision variables are equal to joint velocity commands, the constraints are related to trajectory tracking in (3.86a), velocity in frame satisfying a holonomic constraint in (3.86b), manipulator physical limits in joint angles in (3.86c) and manipulator physical limits in joint velocities in (3.86d), also the objective function is the negative of a manipulability index. Then, the redundant manipulator will track the desired trajectory in its workspace and satisfy the holonomic constraint assuming it has a high position accuracy, the initial end-effector position coincides with the initial trajectory and the velocity ellipsoids defined by the manipulability indexes $w_b(\theta_1, b)$ *and* $w_m(\theta_{b+1,n})$ *are non vanishing in any direction of the task space.*

Proof. See Appendix A.3. □

3.4 Comparison of Methods

The objective of this section is discuss the implementation of each method highlighting their differences.

The Table 3.4 shows the following aspects of each method:

- **References:** The references for kinematic control and QP discuss trajectory tracking problems together with other objectives (for example manipulability and constraints). The references for SQP are the author previously works and the texts that discuss the method.

- **Basic problem formulation:** Inherent from each method.

- **Number of calculations for basic description:** Although the total calculations and the convergence time are problem dependent, among the three methods discussed the SQP is always the slower because it solves a sequence of quadratic problems.

- **Manipulator joint velocities constraints:** The methods QP and SQP define as constraints as part of optimization problem. On the other hand the kinematic control just limits the signal value, which can degrade the method performance.

- **Manipulator joint angle limits:** Again the QP and SQP methods define constraints as part of optimization problem. The kinematic control has to use the null space of Jacobian.

- **Holonomic scleronomic constraint in F_c:** First, for the three methods is necessary to define the scleronomic constraint features: location of in the manipulator kinematic chain, type (displacement or rotating), dimension and value. This way $v_d(t)$ and the matrices $\Phi_{c,b}$, $J(\theta_{1,b})$ and D are defined.

  Both QP and SQP add an equality constraint in the optimization problem. As the QP formulation only supports linear constraints and the information of the sampling period can not be added without some linearization. So, the best scenario to satisfy the scleronomic constraint is run the QP at a high rate.

  On the other hand, the SQP adds information of the joints values (rotation for revolute joints and displacement for prismatic joints) in a sampling period directly in the Jacobian, this way choosing an appropriate sampling period and ensuring a low convergence time of SQP is ideal scenario.

  From the scleronomic constraint features the kinematic control approach determines the constrained Jacobian using it together with a controller (proportional plus feed forward) to find the constrained velocity vector, $[u_f \quad u_{b+1,n}]^T$. Now with this vector, the Jacobian pseudo-inverse of $J_b(\theta_{1,b})$ and the null space of a $D\Phi_c, b$ the control signal that satisfy the scleronomic constraint is found.

  Considering that the QP and SQP algorithms are already coded these methods are more simple to implement than the kinematic control. Also in case of more scleronomic constraints in distinct locations of manipulator kinematic chain just add more equality constraints in the optimization problem formulation of QP and SQP. In contrast, the kinematic control need another batch of calculus probably (formulation still open) including a new Constrained Jacobian matrix.

Lastly, none of the methods add negative feedback information of how much the constraint is far from the desired value.

- **Holonomic rheonomic constraint in F_c:** The comments are the same for the scleronomic constraint, except that $v_d(t)$ is a time dependent function.
- **Maximize an index, for example manipulability:** The SQP is the only method that guarantee total fidelity of the index. The QP needs linearization and the kinematic control needs curve fitting.
- **Stability:** For the kinematic control is proven. QP and SQP need conditions and/or assumptions.

Table 3.4: Comparison among methods for trajectory tracking.

Method	**Kinematic Control**	**Optimization Problem via Quadratic Programming**	**Optimization Problem via Sequential Quadratic Programming**
References	The main reference is the dissertation [16], other works are [17, 25, 26, 63]	The main reference is [21], others works are [43, 84, 85]	The texts that describe the method [41, 56] and author previous works [10, 11].
Basic problem formulation	$\begin{aligned} u &= J^{\dagger}(\theta)u_p \\ u_p &= K_p e_p + \dot{p}_d(t) \\ e_p &= p_d - p_e \end{aligned}$	$\min_u \frac{1}{2}u^T C u + c^T u \in \mathbb{R}$ $\mathcal{F} = \begin{cases} A_i^T u = s_i, & i \in \mathcal{E}; \\ A_i^T u < s_i, & i \in \mathcal{I}; \\ u \in \mathcal{U}, \end{cases}$	$\min_u f(u) \in \mathbb{R}$ $\mathcal{F} = \begin{cases} g_i(u) = 0, & i \in \mathcal{E}; \\ g_i(u) \leq 0, & i \in \mathcal{I}; \\ u \in \mathcal{U}, \end{cases}$
Number of calculations for basic description	Low, a matrix pseudoinverse and a derivative are the most complex calculations.	Medium, generally polynomial time.	High, it solves a sequence of linearized quadratic optimization subproblems.

Continued on the next page

Method	Kinematic Control	Optimization Problem via Quadratic Programming	Optimization Problem via Sequential Quadratic Programming
Manipulator joint velocities constraints	Limit the joint velocities send to robot controller: $$\text{if} \quad u_i > \theta_i^+ \rightarrow u_i = \theta_i^+$$ $$\text{if} \quad u_i < \theta_i^- \rightarrow u_i = \theta_i^-$$	Add inequality constraints: $$\dot{\theta}^- \leq u \leq \dot{\theta}^+$$	Add inequality constraints: $$\dot{\theta}_i^- \leq u_i \leq \dot{\theta}_i^+$$
Manipulator joint angle limits	Use the null space of the Jacobian in the control law: $$u = J^\dagger(\theta)u_p + (I - J^\dagger(\theta)J(\theta))\varphi$$ $$\varphi = K_0 \left(\frac{\partial\beta(\theta)}{\partial\theta}\right)^T$$ $$\theta^- \leq \theta_i \quad \leq \quad \theta^+$$ $$\beta(\theta) = -\frac{1}{2n}\sum_{i=1}^{n}\left(\frac{\theta_i - \theta_i^+/2 - \theta_i^-/2}{\theta_i^+ - \theta_i^-}\right)^2$$	Add inequality constraints: $$\theta^- \leq \theta \leq \theta^+$$	Add inequality constraints: $$\theta_i^- \leq \theta_i + u_i T \leq \theta_i^+$$

Continued on the next page

Method	Kinematic Control	Optimization Problem via Quadratic Programming	Optimization Problem via Sequential Quadratic Programming
Holonomic scleronomic constraint in F_c	Use the constrained Jacobian to find the control degree of freedom as a way of satisfying the constraint: $$D\Phi_{c,b}J_b(\theta_{1,b})u_{1,b} = 0$$ $$\begin{bmatrix} u_f \\ u_{b+1,n} \end{bmatrix} = {J_r}^{\dagger}(\theta_{b+1,n})u_p$$ $$u_{1,b} = {J_b}^{\dagger}(\theta_{1,b})\mathsf{N}(D\Phi_{c,b})u_f$$	Add a equality constraint: $$D\Phi_{c,b}J_b(\theta_{1,b})u_{1,b} = 0$$	Add a equality constraint: $$D\Phi_{c,b}J_b(\theta_{1,b} + u_{1,b}T)u_{1,b} = 0$$

Continued on the next page

Method	Kinematic Control	Optimization Problem via Quadratic Programming	Optimization Problem via Sequential Quadratic Programming
Holonomic rheonomic constraint in F_c	Use the constrained Jacobian to find the control degree of freedom as a way of satisfying the constraint: $$D\Phi_{c,b}J_b(\theta_{1,b})u_{1,b} = v_d(t)$$ $$\begin{bmatrix} u_f \\ u_{k+1,n} \end{bmatrix} = {J_r}^{\dagger}(\theta_{b+1,n})u_p$$ $$u_{1,b} = {J_b}^{\dagger}(\theta_{1,b})\left(\mathsf{N}(D\Phi_{c,b})u_f + (D\Phi_{c,b})^{\dagger}v_d(t)\right)$$	Add a equality constraint: $$D\Phi_{c,b}J_b(\theta_{1,b})u_{1,b} = v_d(t)$$	Add a equality constraint: $$D\Phi_{c,b}J_b(\theta_{1,b} + u_{1,b}T)u_{1,b} = v_d(t)$$

Continued on the next page

Method	**Kinematic Control**	**Optimization Problem via Quadratic Programming**	**Optimization Problem via Sequential Quadratic Programming**
Maximize an index, for example manipulability	Use the null space of the Jacobian in the control law: $$u = J^{\dagger}(\theta)u_p + (I_n - J^{\dagger}(\theta)J(\theta))\varphi$$ $$\varphi = K_0 \left(\frac{\partial w(\theta)}{\partial \theta}\right)^T$$ $$w(\theta) = \sqrt{det(J(\theta)J^T(\theta))}$$	Linearize the manipulability rewriting the objective function as: $$\min_u \quad -\frac{1}{2}u^T H_w(\theta)u - \nabla_w{}^T(\theta)u,$$ and set the following constraint: $$J_A(\theta)u = \dot{p}_d(t)$$	Rewrite the objective function with the manipulability: $$\min_u -w(\theta + uT),$$ and set the trajectory error as a constraint: $$p_d(t+T) - \mathsf{FK}(\theta + uT) = 0$$
Stability	Almost globally asymptotic stability for closed loop system.	With a positive definite $C \in \mathbb{R}^{n\times n}$ its possible to reach a exact solution under certain conditions or at least improve monotonically the actual solution.	Only local convergence, global convergence only under conditions or assumptions.

Chapter 4

Simulation and Experimental Results

This chapter shows simulations and experiments with three methods described in chapter 3, kinematic control, quadratic programming and sequential quadratic programming. All experiments are performed in a Baxter research robot. The Section 4.1 presents a kinematic description of the Baxter research robot while 4.2 shows a preliminary experiment where only trajectory tracking is taken into account (no manipulability or constraints). Section 4.3 presents a manipulability analysis using the indexes defined in Section 2.6. Simulation and experiment regarding a scleronomic constraint are in Sections 4.4 and 4.5, respectively, while the experiment regarding a rheonomic constraint is in Section 4.6.

The trajectory tracking problem in Chapter 3 was formulated considering the manipulator end-effector pose (position plus orientation). For all experiments and simulation the end-effector orientation is despised and only position is considered, properly represented by means of a selection matrix $S \in \mathbb{R}^{3\times 6}$ defined as:

$$S = \begin{bmatrix} I_3 & 0_{3,3} \end{bmatrix}, \tag{4.1}$$

this selection matrix premultiplies the $J_b(\theta_{1,b})$, $J_e(\theta)$ and $\mathsf{N}(D\Phi_{c,b})$ in (2.68), (2.69), (3.28b) and (3.84b). Also in the formulation of kinematic control, QP and SQP, is necessary adjust the size of D and $\Phi_{c,b}$ for $\mathbb{R}^{1,3}$ and $\mathbb{R}^{3,3}$, respectively.

A performance index [20] is a quantitative measure of the system performance and is chosen to emphasis important system specifications. The following indexes are defined in relation to trajectory error being $t_f \in \mathbb{R}$ the task execution time, all integrals are implemented using the trapezoidal rule.

- ISE - integral of the square error. This index discriminate between excessively overdamped and underdamped systems, also is mathematically convenient for analytical purposes:

$$ISE = \int_0^{t_f} e_p^2(t)dt. \tag{4.2}$$

- IAE - integral of the absolute error. This index is particularly useful for computer simulation studies:

$$IAE = \int_0^{t_f} |e_p(t)|\, dt. \tag{4.3}$$

- ITAE - integral of the time multiplied by absolute error. This index reduces the contribution of large initial error and emphasize errors occurring later in response:

$$ITAE = \int_0^{t_f} t\, |e_p(t)|\, dt. \tag{4.4}$$

- ITSE - integral of time multiplied by the squared error. This index has a time-weighted nature and a frequency domain equivalent index the *D-product*, that can be interpreted as a pseudo norm [12].

$$ITSE = \int_0^{t_f} e_p^2(t)dt. \tag{4.5}$$

- l_2 *norm*. This index is a distance measure from the origin of the vector space.

$$l_2\ norm = \parallel e_p \parallel = \sqrt{e_p^T e_p}. \tag{4.6}$$

4.1 Kinematic Model of Baxter Research Robot

In this section the Baxter robot, from manufacturer Rethink Robotics, is described. A geometrical analysis is done in order to obtain the descriptive parameters for the robot kinematic chain. The Baxter robot, Figure 4.1, is a dual arm anthropomorphic robot used originally for simple industrial jobs as loading, unloading, sorting and handling of materials. There are two models, the Baxter industrial robot and the Baxter research robot. The Baxter industrial robot can be programmed moving its hands to perform the desired task, in this way the robot will memorize the movement and be able to repeat the task continuously. With this feature, the Baxter industrial robot is not programmed by engineers writing code, then any regular person with no knowledge of programming and robotics can teach Baxter industrial robot to perform tasks in minutes.

On the other hand, the Baxter research robot is designed to be programmed through the robot operating system (ROS). ROS is a collection of software frame-

Figure 4.1: Baxter® robot used in experiments.

works that provide hardware abstraction, communications infrastructure, robot geometry, among other things. An introduction to ROS is found in [57]. The manufacturer provides a software development system (SDK), using ROS, that offers as key features the communication with a Linux workstation, measurements from position, velocity and torque from joints as also three main modes to control the robot: desired position, actual velocity or effort torque. To develop this work the choice is the Baxter research robot because it is possible read the sensors and command the actuators through ROS.

In order to create a kinematic model for Baxter two steps are performed. First, a geometrical analysis. Second, analysis of the Universal Robotic Description Format (URDF) file generated from Baxter robot in Figure 4.1 with definition of the kinematic model via the homogeneous transformation matrix.

4.1.1 Geometric Analysis

The Baxter robot consists in a fixed torso with two arms and a rotational head, as represent in Figure 4.2. Each arm has 7 revolute joints and each joint has 1 DOF, so an arm has a total of 7 DOF. The manufacturer has its own nomenclature for the joints, namely $s0$, $s1$, $e0$, $e1$, $w0$, $w1$ and $w2$, where the letter s refers to shoulder, the letter e refers to elbow and the letter w refers to wrist. Figure 4.2 shows the location of the joints in the Baxter's right arm as well as the link names connecting these joints.

The Table 4.1 determines the joints manufacturer nomenclature and the respective joint angle. Table 4.1 also presents some physical characteristics of joints, the angle limits in radian (rad), maximum absolute value of velocity in radian per second (rad/s) and peak torque in Newton meter (Nm). The joint velocity can be positive or negative depending on direction of joint rotation. The manufacturer nomenclature is neglected in favor of the nomenclature already presented in Chapter 2. F_i is

Figure 4.2: The arms of Baxter robot, from [38].

Table 4.1: Parameters of Baxter.

Joint	θ_i	**Angle limits** (rad)	|**Maximum velocity**| (rad/s)	**Peak torque** (Nm)
$s0$	θ_1	-2.46 to 0.89	2.0	50
$s1$	θ_2	-2.15 to 1.05	2.0	50
$e0$	θ_3	-3.03 to 3.03	2.0	50
$e1$	θ_4	-0.05 to 2.62	4.0	50
$w0$	θ_5	-3.06 to 3.06	4.0	15
$w1$	θ_6	-1.57 to 2.09	4.0	15
$w2$	θ_7	-3.06 to 3.06	4.0	15

the i-th frame in Baxter's kinematic chain and the i-th frame is attached to the i-th joint.

4.1.2 URDF Analysis

The unified robot description format is the file in a standardized extensible markup language (XML) that describes a robot model detailing its parts, joints, dimensions and other features. Details of URDF can be found in [37, 50] and details of XML can be found in [65]. Using the URDF Baxter's file the URDF diagram of Baxter kinematic model is obtained. Figure 4.3 shows part of the URDF Baxter diagram (on the left) and the XML code (on the right). The frames (as also the joints) are the ellipses and the links are the rectangles. Joints and links are connected by arrows. The nomenclature xyz followed by numbers is the distance in m (meters) between two frames in axes x (first number), y (second number) and z (third number) in the body frame (frame that the link of the rectangle is on). The nomenclature rpy followed by numbers is the orientation between two frames in rad considering RPY angles, where first number is the roll angle, second number is the pitch angle

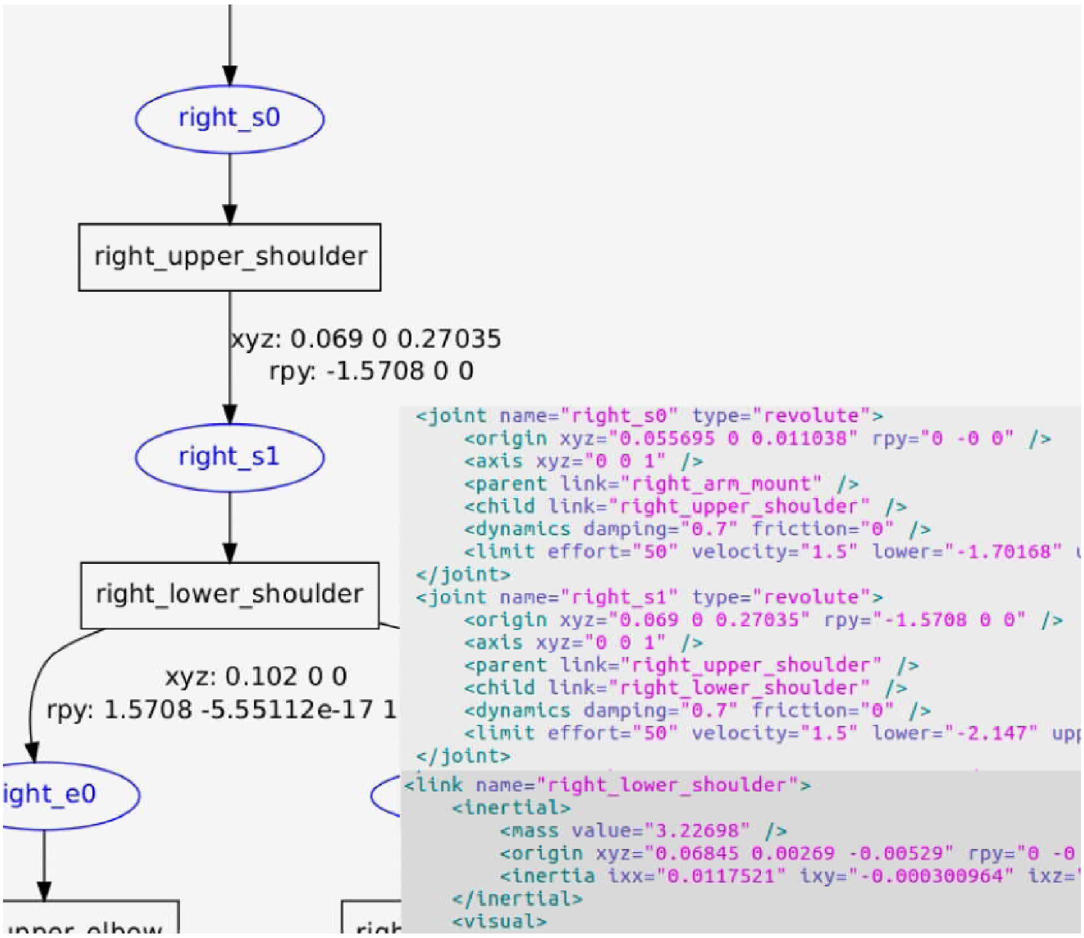
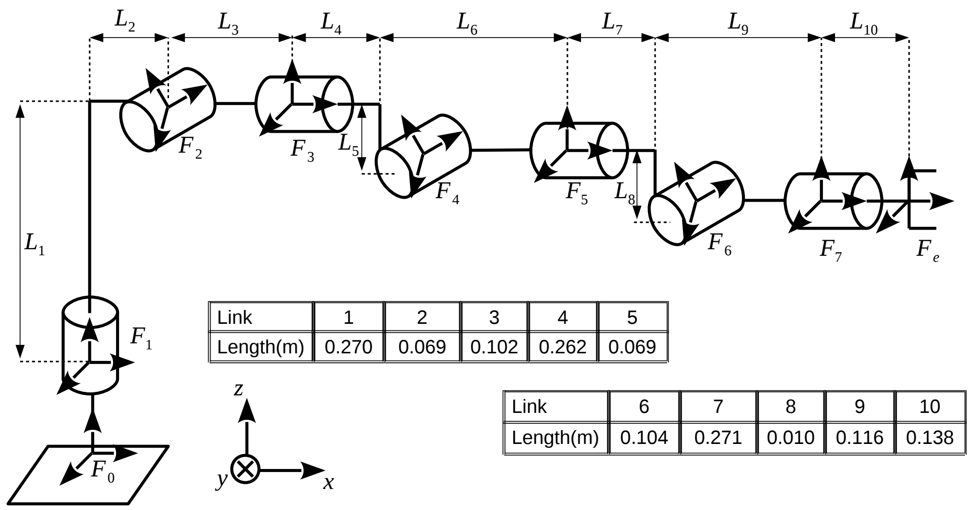


Figure 4.3: Part of URDF diagram of Baxter and XML code.

and third number is the yaw angle. The XML code is explained by itself: the nomenclature $< tag >$ is the beginning of a tag and $< /tag >$ the end, and there is also a short notation $< tag = data/ >$. The parameters of a tag are always inside the tag beginning and tag end.

Based on the Baxter's URDF file, a simplified kinematic representation of the Baxter's right arm is created, Figure 4.4. In this figure, L_i is the distance in meters along the axis between two joints, j_i represents a revolute joint which is located in the respective frame F_i. From now on the calculations consider only the Baxter's right arm.

Now it is possible to determine the homogeneous transformation matrices of Baxter robot using the parameters from Figure 4.4, except $T_{0,1}$ (obtained directly from the URDF file). So, the homogeneous transformation matrix from F_1 to inertial



Link	1	2	3	4	5
Length(m)	0.270	0.069	0.102	0.262	0.069

Link	6	7	8	9	10
Length(m)	0.104	0.271	0.010	0.116	0.138

Figure 4.4: Kinematic model of Baxter's right arm. All L_i are in meters, in each revolute joint j_i is located the respective frame F_i.

frame F_0 is given by:

$$T_{0,1} = \begin{bmatrix} \cos(-\pi/4) & -\sin(-\pi/4) & 0 & 0.064 \\ \sin(-\pi/4) & \cos(-\pi/4) & 0 & -0.259 \\ 0 & 0 & 1 & 0.129 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.7}$$

The compound homogeneous transformation from F_e to F_0 in the Baxter's arm is given by:

$$T_{0,e} = T_{0,1}T_{1,7}T_{7,e} = \begin{bmatrix} R_{0,e} & (r_{0,e})_0 \\ 0_{1,3} & 1 \end{bmatrix}, \tag{4.8}$$

where $(r_{0,e})_0 \in \mathbb{R}^3$ and $R_{0,e} \in \mathbb{R}^{3\times 3}$ provide the end-effector position and orientation in the inertial frame, respectively.

4.2 Preliminary Experiment

The communication with Baxter is performed using the scheme of Figure 4.5. A computer is connected via an Ethernet cable directly into Baxter. The computer needs an Ubuntu operating system together with the ROS framework (needs also some ROS packages coded by the Baxter's manufacturer), the versions of Ubuntu and ROS depend on the Baxter installed firmware. The code can be written on either python (interpreted language) or C++ (compiled language), for this thesis the choice was python because generally it is easier to debug an interpreted language. The Baxter uses the Gentoo operating system but this can be abstracted by the user unless some kind of maintenance needs to be executed.

Figure 4.5: Experimental configuration.

The computer used for the experiments uses Ubuntu 12.04 LTS operating system, ROS Groovy distribution together with version 2.7 of Python. It has an Intel core i7-5500U 2.40 GHz processor, an Intel HD 5500 onboard video card and 8 GB DDR3 RAM memory. The experiments and simulation regarding the QP in uses the package CVXOPT (Python programming language implementation) [78] with the cone QP method. The experiments and simulation regarding the SQP uses the package pyOPT (Python programming language implementation) [61] with the SLSQP method.

A preliminary experiment is defined as follows: the Baxter end-effector tracks a desired trajectory and there is no holonomic constraint in Baxter kinematic chain. The objective of this preliminary experiment is to confirm that the environment (communications, packages, code) is settled and the three methods considered (KC, QP and SQP) are able to drive the robot.

The desired end-effector trajectory for the preliminary experiment is given by:

$$p_d(t) = \begin{bmatrix} p_x(0) + 15\sin(\pi t/20) \\ p_y(0) + 45\sin(2\pi t/20) \\ p_z(0) + 30\sin(2\pi t/20) \end{bmatrix} mm, \tag{4.9}$$

where $p_x(0)$, $p_y(0)$ and $p_z(0)$ are the initial positions at natural basis for a Euclidean three-dimensional space in axes x, y and z, respectively. The initial state of the joint angles and the initial end-effector position are defined in Table 4.2 and Table 4.3, respectively. The task execution time is 40 s. The desired trajectory in (4.9) considering the values given by Tables 4.2 and 4.3 is visualized in Figure 4.6.

For all experiments and simulation the same following parameters ares set. In

Table 4.2: Initial state of the joint angles for the desired trajectory defined in (4.9).

Joint angle	Value (rad/s)
θ_1	$\pi/6$
θ_2	$-\pi/6$
θ_3	$\pi/3$
θ_4	$\pi/4$
θ_5	$-\pi/3$
θ_6	$\pi/4$
θ_7	0

Table 4.3: Initial end-effector position for the desired trajectory defined in (4.9).

Axis	Value (mm)
p_x	1071
p_y	-109
p_z	326

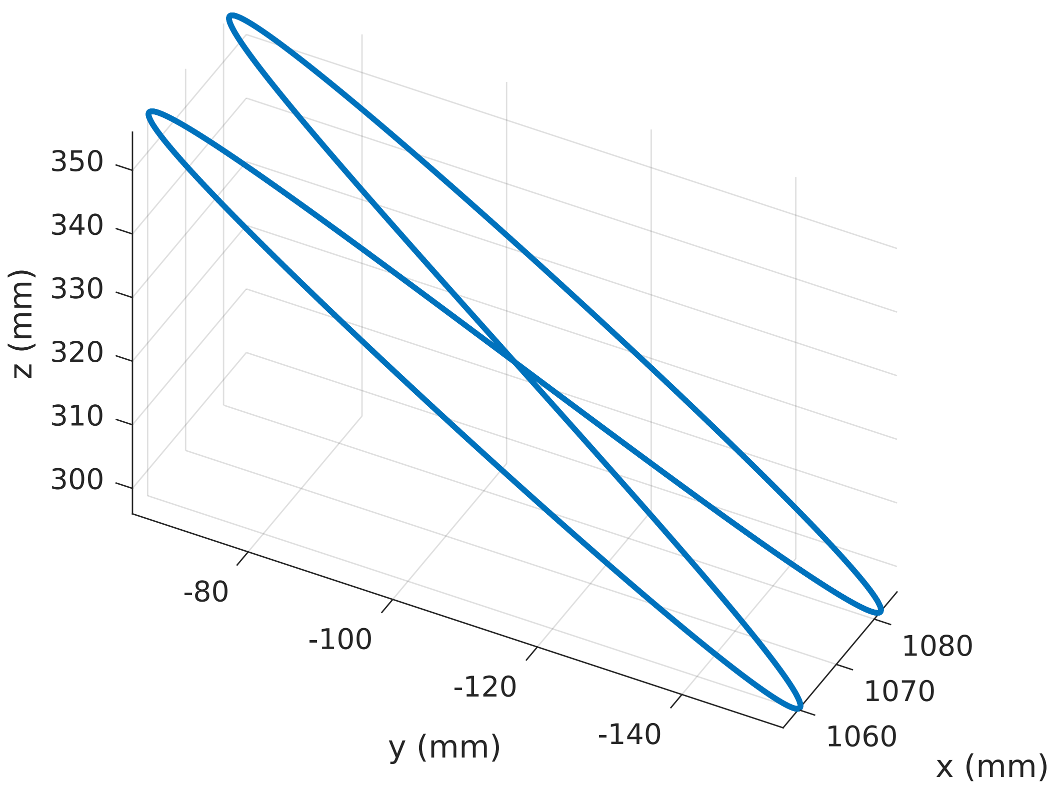


Figure 4.6: Desired trajectory defined in (4.9).

Table 4.4: Performance indexes, preliminary experiment.

Index	KC	QP	SQP
ISE e_x	6.33e-06	2.85e-05	1.41e-05
ISE e_y	1.40e-04	3.35e-04	2.31e-04
ISE e_z	2.40e-04	3.04e-04	2.32e-04
IAE e_x	1.18e-02	2.40e-02	1.84e-02
IAE e_y	5.93e-02	8.74e-02	7.87e-02
IAE e_z	7.21e-02	8.83e-02	7.49e-02
ITAE e_x	2.34e-01	4.08e-01	3.40e-01
ITAE e_y	1.24e+00	1.60e+00	1.51e+00
ITAE e_z	1.39e+00	2.01e+00	1.56e+00
ITSE e_x	1.31e-04	3.94e-04	2.47e-04
ITSE e_y	3.00e-03	5.38e-03	4.24e-03
ITSE e_z	4.26e-03	7.62e-03	4.87e-03
$\|\| e_x \|\|$	1.13e-02	4.75e-02	1.68e-02
$\|\| e_y \|\|$	5.29e-02	1.66e-01	6.80e-02
$\|\| e_z \|\|$	6.92e-02	1.54e-01	6.81e-02

kinematic control the gain matrix has a constant value $K_p = diag(2.5, 3.0, 3.75)$ (where $diag()$ is a diagonal matrix) and the sampling period $T = 0.05\ s$, in QP the gain $k_1 = 6.0$ and $T = 0.012\ s$, in SQP $T = 0.05\ s$.

The Baxter is a robot with low position accuracy [33] because the hardware limits and the existence of series elastic actuator (SEA) [64] in its joints. In this way, the trajectory errors for the preliminary test in Figure 4.7 are expected to present some variation about $\pm 5\ mm$ (manufactures published accuracy) even with no holonomic constraints in the kinematic chain.

Figure 4.7 shows that the error on the x axis is the smallest for the three methods, always below 5 mm. KC and SQP had worse results on the z axis with some values around 10 mm while QP showed some values around 10 mm for both z axis and x axis.

Table 4.4 shows the performance indexes for the preliminary experiment. KC achieved the best results except for ISE and l_2 $norm$ on the z axis. The QP had the worst results in all indexes, but was not an order of magnitude above the best result in any index. In general, SQP results were intermediate, sometimes closer to KC or QP.

Figure 4.8 shows the joint control signals for the preliminary experiment. KC has the lowest amplitude QP the largest. In KC the variation of the signals is milder compared to the more aggressive variation of QP and SQP.

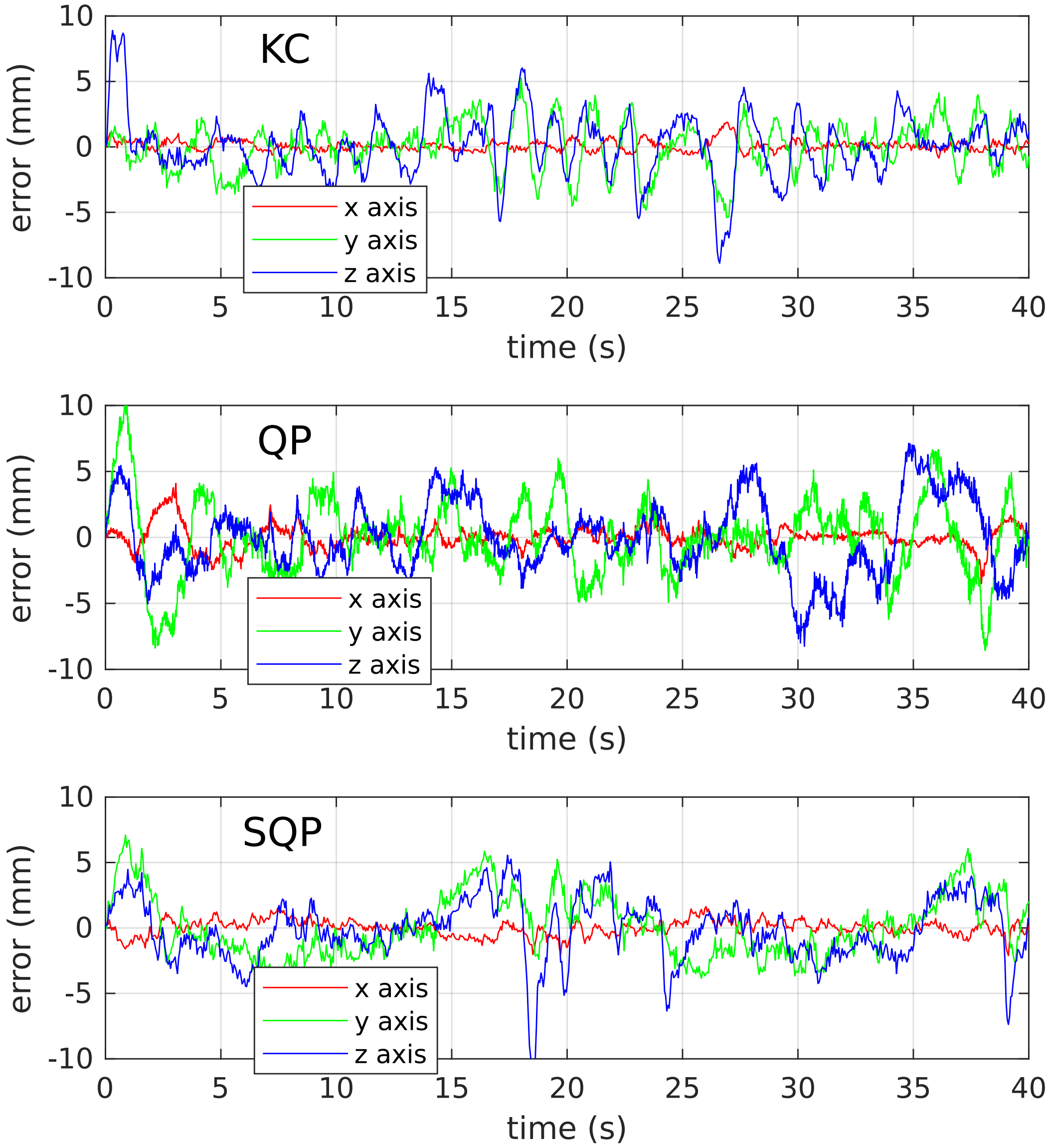


Figure 4.7: Trajectory tracking error for preliminary experiment.

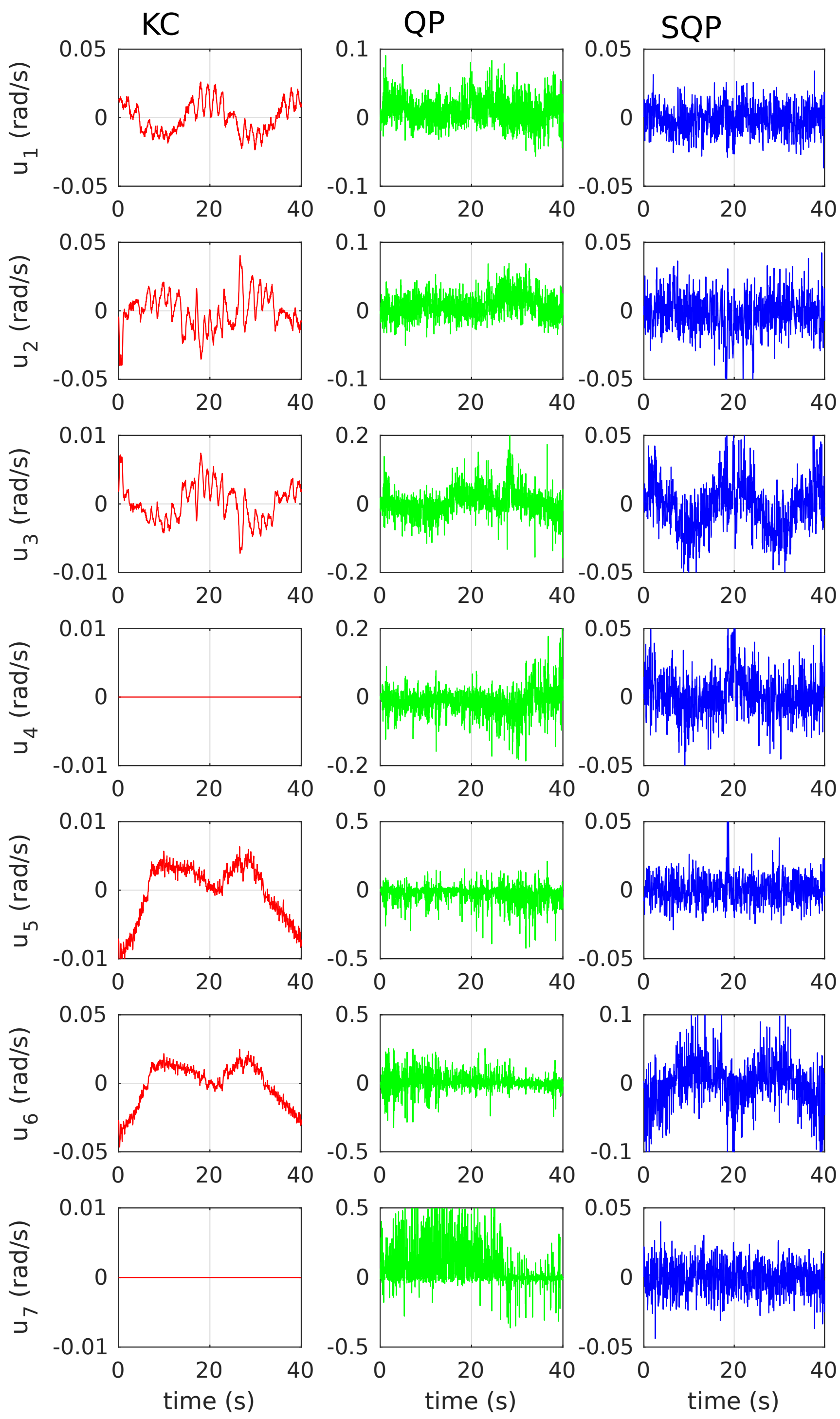


Figure 4.8: Joint control signals, preliminary experiment.

4.3 Baxter Manipulability

For the experiments e simulation considering a constraint in Baxter's kinematic chain, manipulability and trajectory tracking, the holonomic constraint is defined arbitrary in the Baxter's kinematic chain between F_4 and F_5, in this way $b = 4$. The displacement of the holonomic constraint from F_4 also is defined arbitrary by $L_c = 50\ mm$, as can be seen in Figure 4.9. The type of the constraint is a displacement constraint in the x axis of F_c being the matrix D defined by:

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{4.10}$$

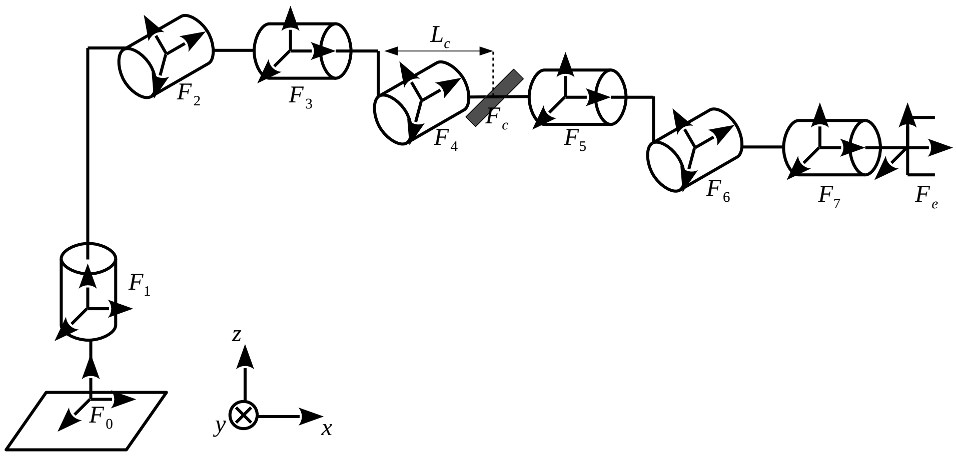


Figure 4.9: Kinematic model of Baxter's right arm with plane constraint between F_4 and F_5.

Figure 4.10 shows $w_b(\theta_1, 4)$ as function of θ_2 and θ_3, a blue color means that the robot is near a singularity and a yellow color means that the robot is far from singular configurations. As the manipulability of $SJ_4(\theta_{1,4})$ takes into account only position until F_4, $w_b(\theta_1, 4)$ does not depend on θ_4 neither θ_1 because in the inertial frame the last column of $SJ_4(\theta_{1,4})$ is null while in the body frame the first column is null, the manipulability value is not affected for frame changes. The singular configuration is reached when $\theta_3 = 0$ as also multiple of $\theta_3 = \pm\pi/2$. The variation of θ_2 does not change $w_b(\theta_1, 4)$. High values of $w_b(\theta_1, 4)$ are reached near odd multiples of $\theta_3 = \pm\pi/4$.

Figure 4.11 shows $w_r(\theta_{5,7})$ in function of θ_5 and θ_6. In Baxter, as θ_7 is coupled up to a revolute joint in x axis, it does not change the end-effector position (only orientation), then it does not change the index w_r. Visualization of angle values for singular configurations would be tricky in a 3D plot, so Figure 4.11 shows $w_r(\theta_{5,7})$ in a 2D plot, a dark blue area means the manipulator is near a singular configuration, while yellow area means the manipulability reached a high value.

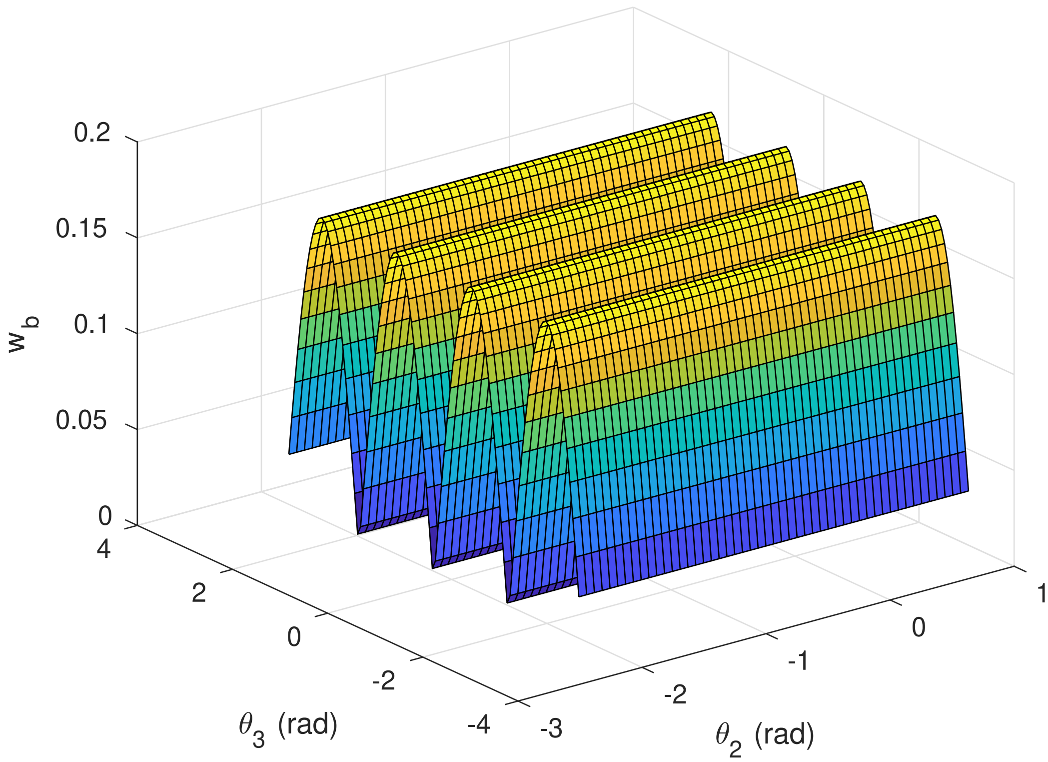


Figure 4.10: Manipulability $w_b(\theta_{1,4})$ with $b = 4$, w_b is multiplied by 10^3.

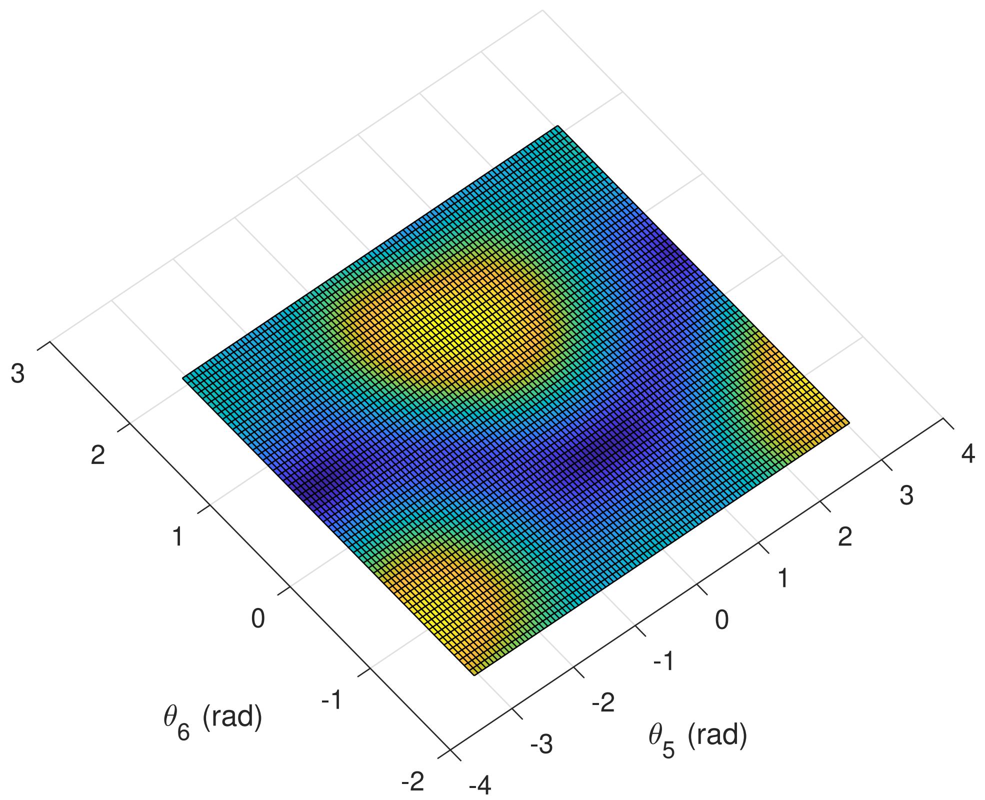


Figure 4.11: Manipulability $w_r(\theta_{5,7})$ in a $\theta_5 - \theta_6$ space with plane constraint between frames F_4 and F_5.

To maximize the manipulability, the kinematic control method uses (3.16) and (3.15). This means that the derivatives of analytical expressions for $\partial w_b(\theta_{1,4})/\partial\theta$ and $\partial w_r(\theta_{5,7})/\partial\theta$ shall be available. These expressions have hundreds of terms making impractical to treat them in a short sampling period. Thus, the curve fitting approach of the manipulability functions is used for the kinematic control. In Figure 4.10, $w_b(\theta_{1,4})$ has a sinusoidal shape, so the following second order Fourier series is used for curve fitting with Figure 4.12 showing the resulting curve:

$$\begin{aligned} w_b(\theta_{1,4}) \approx 1.048\times10^{-4} - 6.922\times10^{-5}\cos(3.994\theta_3) \\ -2.868\times10^{-9}\sin(3.994\theta_3) - 1.333\times10^{-4}\cos(3.994\theta_3) \\ -5.355\times10^{-9}\sin(3.994\theta_3). \end{aligned} \tag{4.11}$$

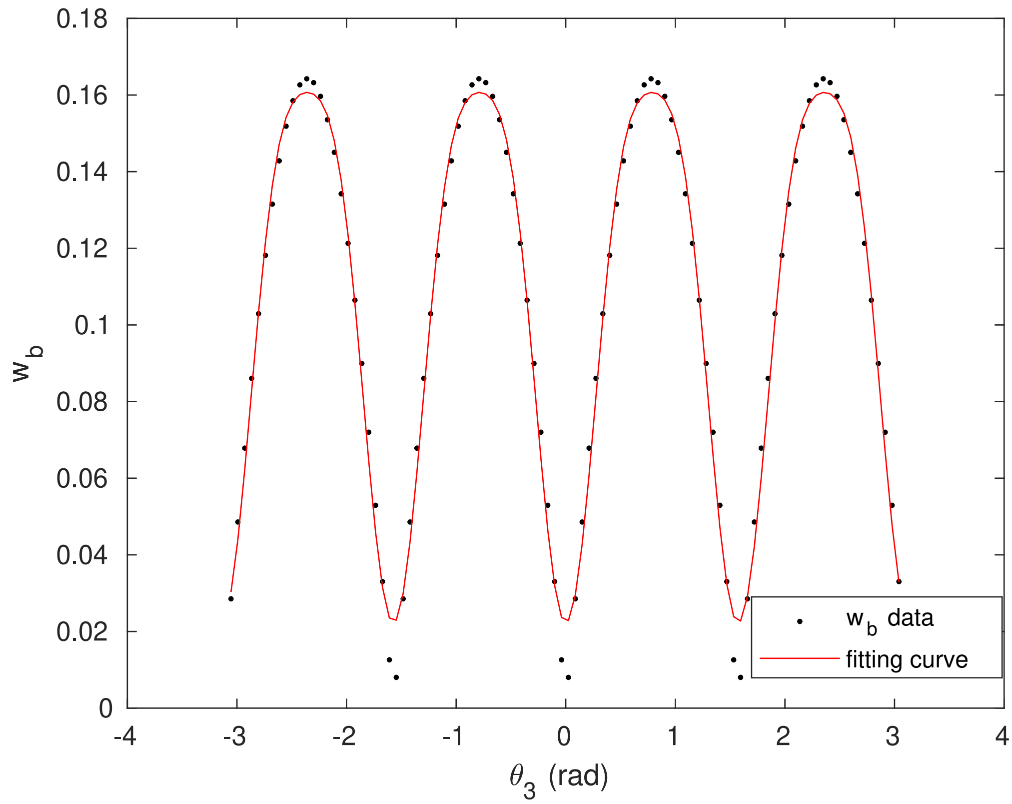


Figure 4.12: $w_b(\theta_1, 4)$ curve fitting, $w_b(\theta_1, 4)$ is multiplied by 10^3.

The approach to curve fitting $w_r(\theta_{5,7})$ is divide the plane $\theta_5 - \theta_6$ in three regions and use one surface in each region, according to Figure 4.13. Depending on θ_5 and θ_6 values, a surface of each region will be used. If $\theta_5 \geq 0$ and $\theta_6 < 0.3911\theta_5 - 0.565$ region 3 is used, else if $\theta_5 < 0$ and $\theta_6 < -0.3911\theta_5 - 0.565$ region 2 is used, else region 1 is used. The surface for each region is defined by a fifth-order polynomial from the

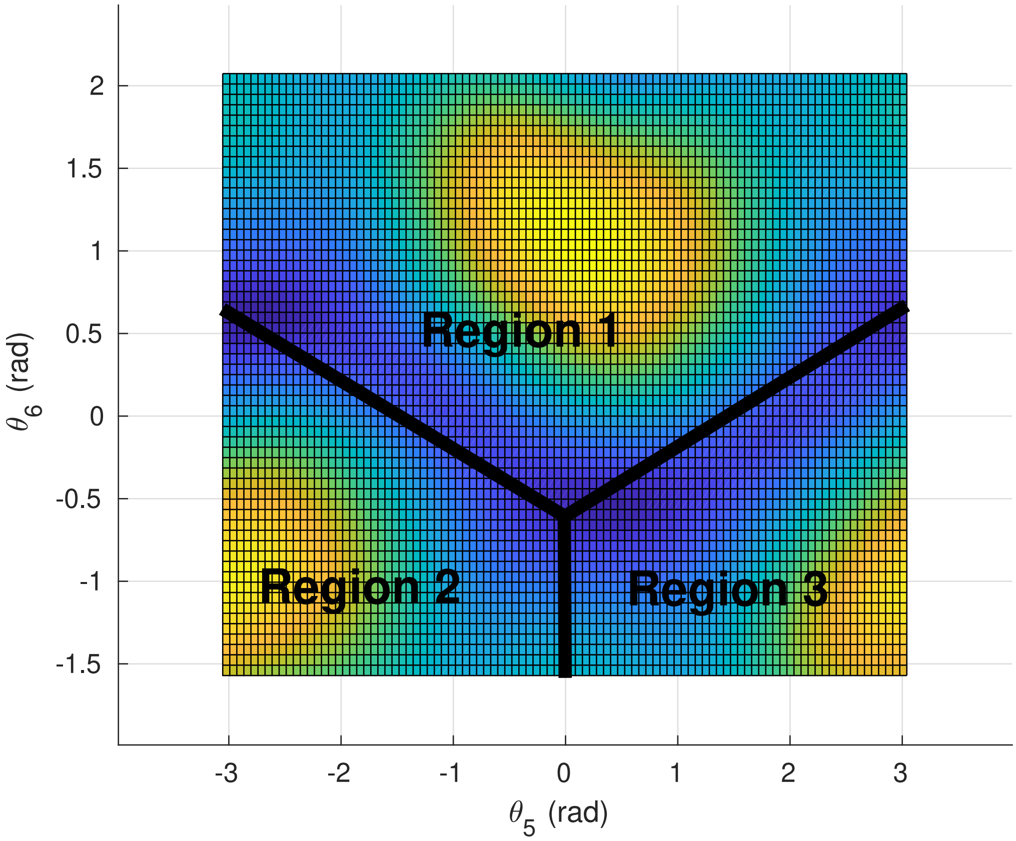


Figure 4.13: Manipulability $w_r(\theta_{5,7})$ in a $\theta_5 - \theta_6$ plane divided in three regions.

following expression with the parameters values $c_{ij} \in \mathbb{R} \;\; i = 0, \ldots, 5; j = 0, \ldots, 5$ in Table 4.3:

$$\begin{aligned} w_r(\theta_{5,7}) \approx\ & c_{00} + c_{10}\theta_5 + c_{01}\theta_6 + c_{20}\theta_5^2 + c_{11}\theta_5\theta_6 + c_{02}\theta_6^2 \\ & + c_{30}\theta_5^3 + c_{21}\theta_5^2\theta_6 + c_{12}\theta_5\theta_6^2 + c_{03}\theta_6^3 + c_{40}\theta_5^4 + c_{31}\theta_5^3\theta_6 \\ & + c_{22}\theta_5^2\theta_6^2 + c_{13}\theta_5\theta_6^3 + c_{04}\theta_6^4 + c_{50}\theta_5^5 + c_{41}\theta_5^4\theta_6 \\ & + c_{32}\theta_5^3\theta_6^2 + c_{23}\theta_5^2\theta_6^3 + c_{14}\theta_5\theta_6^4 + c_{05}\theta_6^5. \end{aligned} \tag{4.12}$$

The QP method approximates the manipulability for a second order function using (3.22) to (3.26) with $\delta = 0.01$. As $w_b(\theta_{1,4})$ depends on θ_3 only ∇w_{b3} and $H_{wb3,3}$ need to be calculated considering other values equal to zero. As $w_r(\theta_{5,7})$ depends on θ_5 and θ_6 only the following values need to be taken into account: ∇w_{r5}, ∇w_{r6}, $H_{wr5,5}$, $H_{wr5,6}$, $H_{wr6,5}$ and $H_{wr6,6}$.

The SQP does not need any approximation or fitting to incorporate the manipulability in the objective function. So in terms of representation of true manipulability value, the SQP has an advantage upon kinematic control and QP methods.

In order to maximize the manipulability the three methods aim to find velocity

Table 4.5: Parameters values for $w_r(\theta_{5,7})$ curve fitting.

Parameter	Region 1	Region 2	Region 3
c_{00}	0.08138	0.1974	0.05468
c_{10}	0.02317	-0.06029	-0.006149
c_{01}	0.1476	0.7356	0.1997
c_{20}	-0.02001	-0.07852	-0.07713
c_{11}	0.03201	-0.3688	0.1848
c_{02}	0.02353	0.9859	0.33
c_{30}	-0.006801	0.04854	-0.08534
c_{21}	-0.06291	-0.05394	-0.02896
c_{12}	-0.07773	-0.4916	0.2412
c_{03}	-0.08908	0.453	0.1595
c_{40}	0.00371	-0.006445	-0.0294
c_{31}	0.007955	0.03075	-0.02699
c_{22}	0.06032	0.03668	0.02751
c_{13}	0.03831	-0.2022	0.0762
c_{04}	0.01848	0.05396	0.02897
c_{50}	-4.334×10^{-5}	-8.346×10^{-5}	-0.003221
c_{41}	-0.001319	-0.001207	-0.003226
c_{32}	-0.001929	0.00685	-0.005685
c_{23}	-0.01301	0.02262	0.01642
c_{14}	-0.005596	-0.0206	0.002887
c_{05}	0.001457	-0.005558	0.001148

commands that result in joint angles translating in the peaks of Figures 4.10 and 4.11. To represent the momentary values of $w_b(\theta_{1,4})$ or $w_r(\theta_{5,7})$ in one index, the integral of the manipulability indexes are taken into account:

$$W_b = \int_0^{t_f} w_b(\theta_{1,4})dt, \tag{4.13}$$

$$W_r = \int_0^{t_f} w_r(\theta_{5,7})dt, \tag{4.14}$$

so, one solution is defined as a pair $W_b \in \mathbb{R}$ and $W_r \in \mathbb{R}$ being classified in dominated or non dominated.

4.4 Simulation with a Scleronomic Constraint

The simulations are performed in Gazebo, an open-source 3D robotics simulator. A picture of Gazebo environment with a Baxter model loaded can be seen in Figure 4.14. The computer used for the simulations uses the Ubuntu 16.04 LTS operating system, ROS Kinetic distribution together with the version 2.7 of Python. The computer has an AMD Ryzen 5 2600x 3.60 GHz processor, an AMD Radeon RX580 8GB DDR5 video card and 16 GB DDR4 RAM memory.

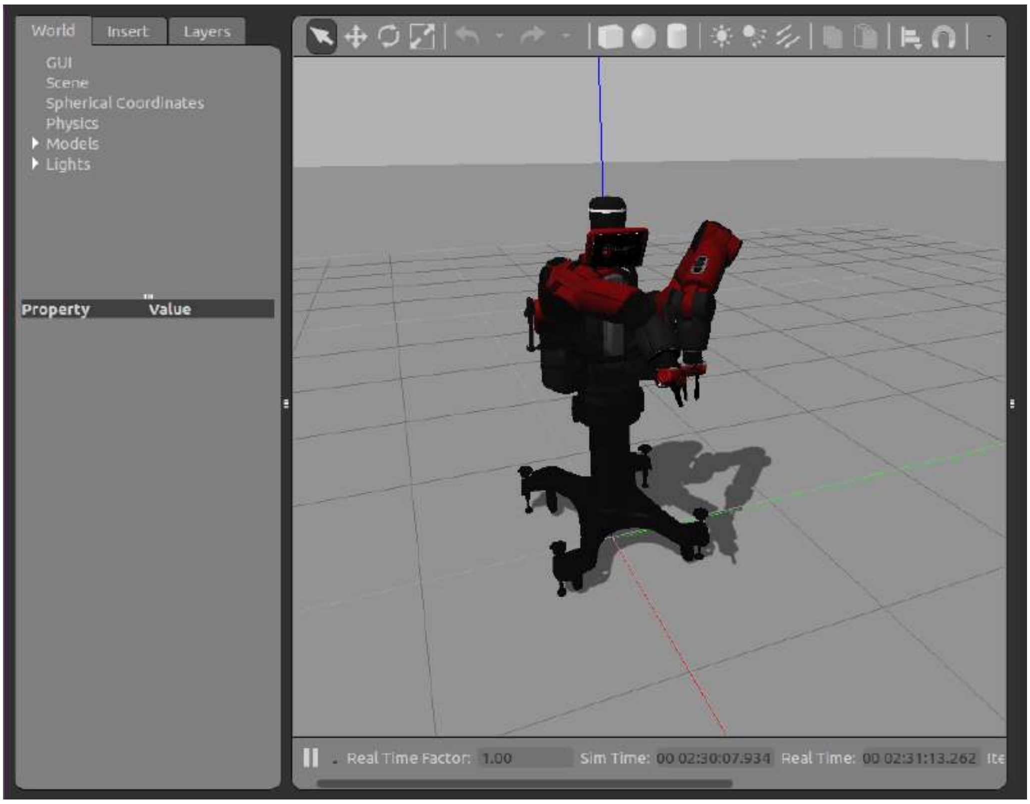


Figure 4.14: Gazebo environment with Baxter model.

Table 4.6: Initial end-effector position for the desired trajectory defined in simulations.

Axis	Value (mm)
p_x	1070
p_y	-114
p_z	328

One important factor about Gazebo is the parameter Real Time Factor, defined by the actual the real time over the simulation time in a window of time, this parameter can be seen in the bottom of Figure 4.14. When simulating the trajectory tacking problem the trajectory is defined in real time, so it is desirable that the Real Time Factor stays close to one during all simulation otherwise the number of samples will be much smaller than an experiment performed on a real robot.

In this simulation the Baxter model has to track the desired trajectory defined by (4.9), the initial state of the joint angles and initial end-effector position are given by Tables 4.2 and 4.6, respectively. The values from Tables 4.3 and 4.6 are not equal because the manipulator in simulation owns a robotic claw, absent in the real robot. So, the values of W_r are expected to be higher in simulations. The shape of desired trajectory is the same as depicted in Figure 4.6, except by offsets in all axes.

Figure 4.15 shows the solution set for the three methods, kinematic control, QP and SQP. In kinematic control the solutions grouped with a $W_r < 4.0$ and $5.5 \times 10^{-3} < W_b < 7.0 \times 10^{-3}$ are defined with a gain $K_r = 0$ and a gain k_b ranging from 0 to 1000. An increase in k_r gain means an increase in W_r and consequently in $w_r(\theta_{5,7})$. There are a total of 38 samples and 5 form the Pareto set. In the QP method, the solutions obtained are poor in magnitude, since the solutions that form the Pareto set are far from the Pareto set solutions of kinematic control and SQP.

In SQP 101 solutions are defined using $\alpha = \begin{bmatrix} 0.00 & 0.01 & \cdots & 0.99 & 1.00 \end{bmatrix}$ and the weighted sum approach. Most solutions with $\alpha < 0.30$ are grouped in the lower right corner of the graph with a high W_b value and a low W_r value. An increase in the α causes an increase in W_r value, but for most solutions it also causes a decrease in W_b value. The best W_b values are obtained by kinematic control and the best W_r values by SQP.

Figure 4.16 shows the trajectory error for some of the highlighted solutions in Figure 4.15. In kinematic control error starts high but in less than 5 seconds the magnitude is already less than 5 mm. The QP presents a considerable variation of the error in terms of the desired point regardless of the analyzed axis. The variation of the SQP is smaller in relation to the QP being the z axis presenting the largest variation.

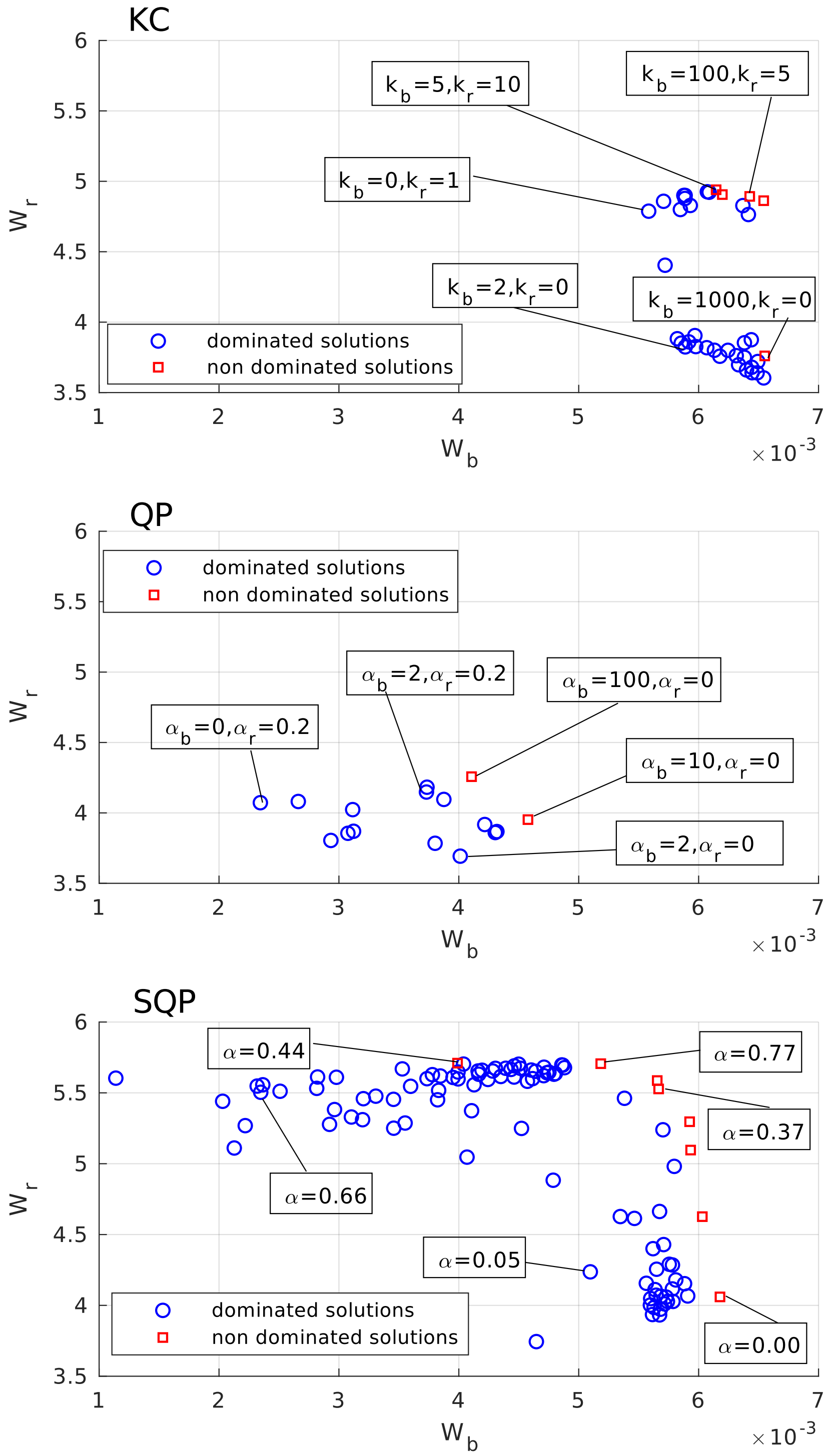


Figure 4.15: W_b and W_r, manipulator satisfy a scleronomic constraint in simulation.

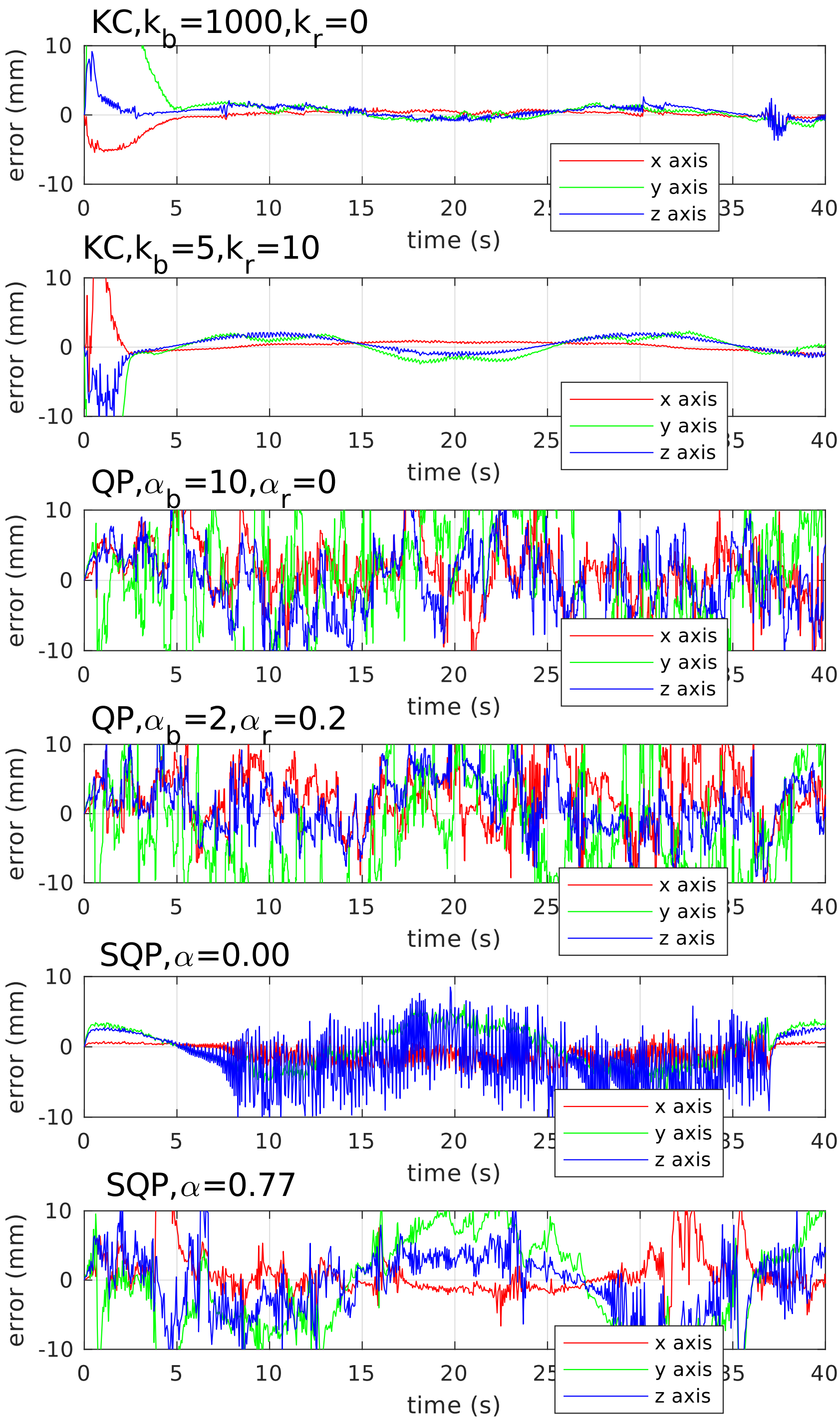


Figure 4.16: Trajectory error, manipulator satisfy a scleronomic constraint in simulation.

Table 4.7: Performance indexes, manipulator satisfy a scleronomic constraint in simulation.

Index	KC $k_b = 1000$ $k_r = 0$	KC $k_b = 5$ $k_r = 10$	QP $\alpha_b = 10$ $\alpha_r = 0$	QP $\alpha_b = 2$ $\alpha_r = 0.2$	SQP $\alpha = 0.00$	SQP $\alpha = 0.77$
ISE e_x	7.37e-05	1.48e-04	9.47e-04	1.14e-03	1.16e-04	7.35e-04
ISE e_y	2.21e-03	4.05e-03	3.27e-03	4.77e-03	2.95e-04	3.02e-03
ISE e_z	6.61e-05	1.38e-04	1.15e-03	8.90e-04	8.15e-04	1.15e-03
IAE e_x	2.93e-02	3.35e-02	1.43e-01	1.61e-01	5.18e-02	9.33e-02
IAE e_y	1.12e-01	1.27e-01	2.88e-01	3.43e-01	9.56e-02	2.71e-01
IAE e_z	3.48e-02	4.93e-02	1.71e-01	1.44e-01	1.44e-01	1.63e-01
ITAE e_x	3.30e-01	4.45e-01	2.69e+00	3.39e+00	1.14e+00	1.67e+00
ITAE e_y	7.19e-01	9.73e-01	6.10e+00	6.71e+00	1.98e+00	5.67e+00
ITAE e_z	6.10e-01	7.89e-01	3.30e+00	2.89e+00	3.09e+00	3.36e+00
ITSE e_x	3.13e-04	4.02e-04	1.74e-02	2.54e-02	2.66e-03	9.89e-03
ITSE e_y	3.37e-03	4.89e-03	7.13e-02	8.63e-02	6.20e-03	5.96e-02
ITSE e_z	7.17e-04	1.11e-03	2.10e-02	1.80e-02	1.82e-02	2.57e-02
$\parallel e_x \parallel$	3.83e-02	5.44e-02	2.97e-01	3.14e-01	4.81e-02	1.21e-01
$\parallel e_y \parallel$	2.10e-01	2.84e-01	5.50e-01	6.59e-01	7.67e-02	2.45e-01
$\parallel e_z \parallel$	3.63e-02	5.25e-02	3.28e-01	2.79e-01	1.27e-01	1.51e-01

Table 4.4 shows the performance indexes for the trajectory errors of Figure 4.16. In general the best results are from kinematic control and the worst from PQ. In relation to the ITSE, the kinematic control is an order of magnitude smaller than the QP. In ISE, SQP has better results with $\alpha = 0.0$ on x and y axes than kinematic control with $k_b = 5, k_r = 10$. In ITAE, only the kinematic control has magnitude less than 1.00 indicating low variability near the end of the trajectory.

The velocity in the constraint is depicted in Figure 4.17, v_c is the constraint velocity in frame F_c while v_d is the desired velocity. In kinematic control the velocity has a sinusoidal shape, the initial amplitudes are high and the smallest variability is obtained with $k_b = 5, k_r = 10$. In QP, although the velocity average value is close to zero, the amplitude peak reaches values around 40 mm/s. In SQP for $\alpha = 0.0$ the velocity average value is below zero and for $\alpha = 0.0$ the initial variation is high with values exceeding 10 mm/s.

Figure 4.18 shows the evolution of manipulability indexes. In the kinematic control $w_b(\theta_{1,4})$ always has a high value, with $k_r = 0$ the $w_r(\theta_{5,7})$ remains practically constant throughout the trajectory while for $k_r = 10$ $w_r(\theta_{5,7})$ increases at the beginning of the trajectory and then stays almost constant. In the QP method, $w_r(\theta_{5,7})$ remains virtually constant on both graphs while $w_b(\theta_{1,4})$ reaches zero at about half of the trajectory for $\alpha_b = 10$ and at the end of the trajectory for $\alpha_b = 10$. In SQP

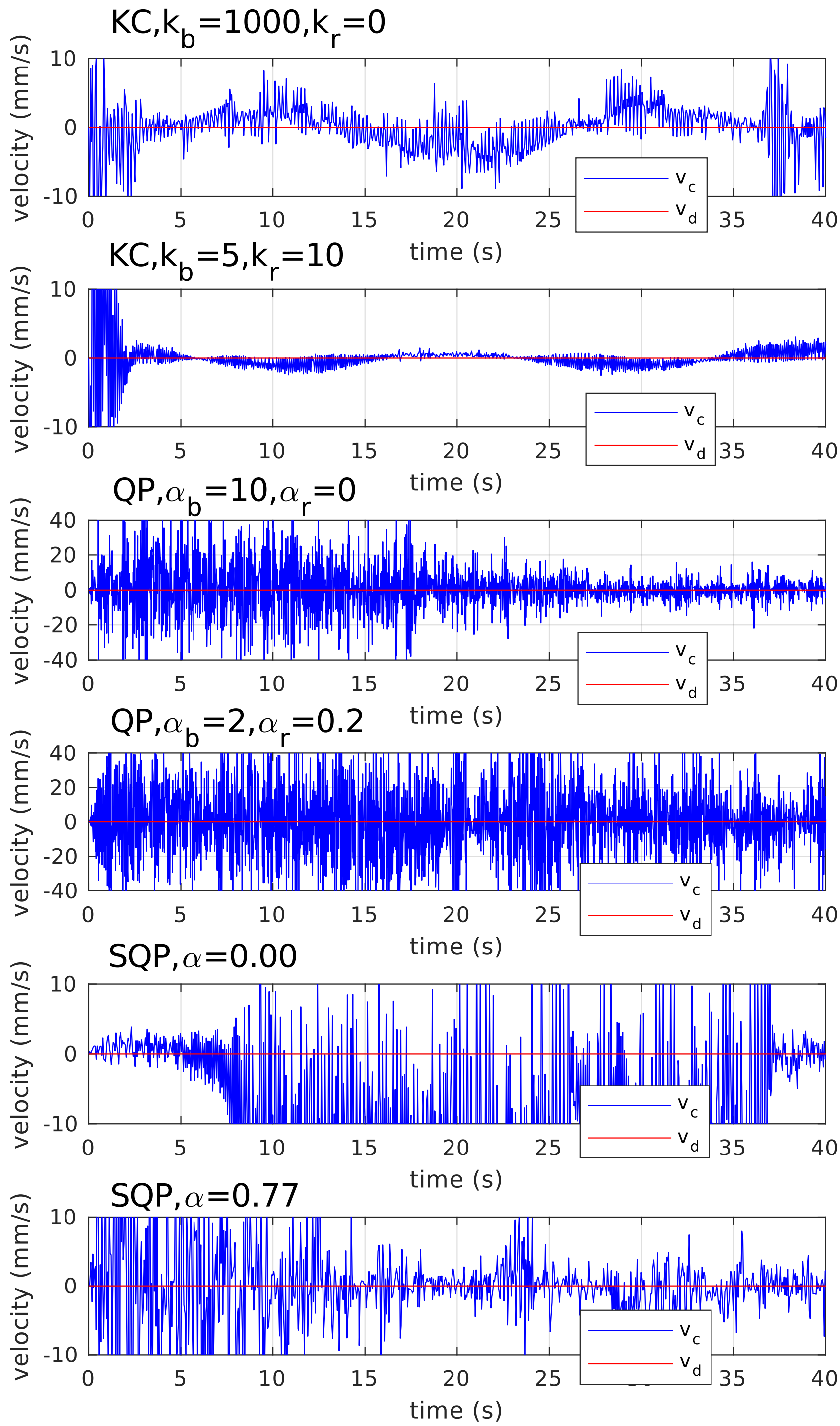


Figure 4.17: Velocity in the constraint, manipulator satisfy a scleronomic constraint in simulation.

for $\alpha = 0.0$, $w_b(\theta_{1,4})$ has a tendency to increase along the trajectory and $w_r(\theta_{5,7})$ remains almost constant, while for $\alpha = 0.77$, $w_b(\theta_{1,4})$ falls below 0.1×10^{-3} to increase later and $w_r(\theta_{5,7})$ increases up to half the trajectory to then remain virtually constant.

Figures 4.19 and 4.20 show the control signal for all Baxter joints. The kinematic control has the smallest amplitudes and the smoothest curves, some in sinusoidal shape, despite wide variations at the beginning of the trajectory. The QP presents signals with abrupt variations, in noise format considering the 40 seconds window, with amplitudes exceeding 2 rad/s. In SQP, signals also show abrupt variations but with a much smaller amplitude than QP, for $\alpha = 0.77$ the maximum amplitudes are greater than for $\alpha = 0.00$.

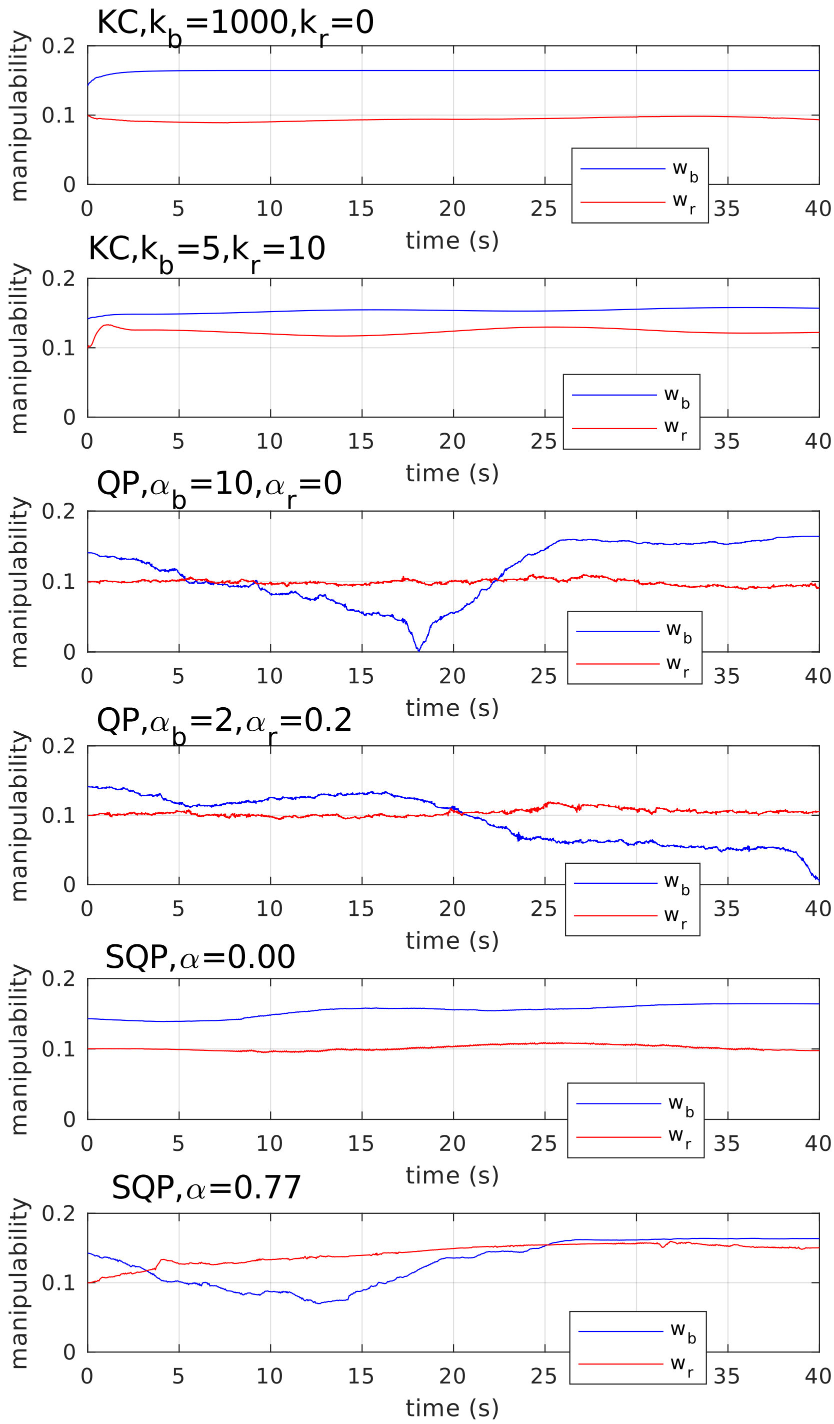


Figure 4.18: Manipulability indexes w_b and w_r (w_b is multiplied by 10^3), manipulator satisfy a scleronomic constraint in simulation.

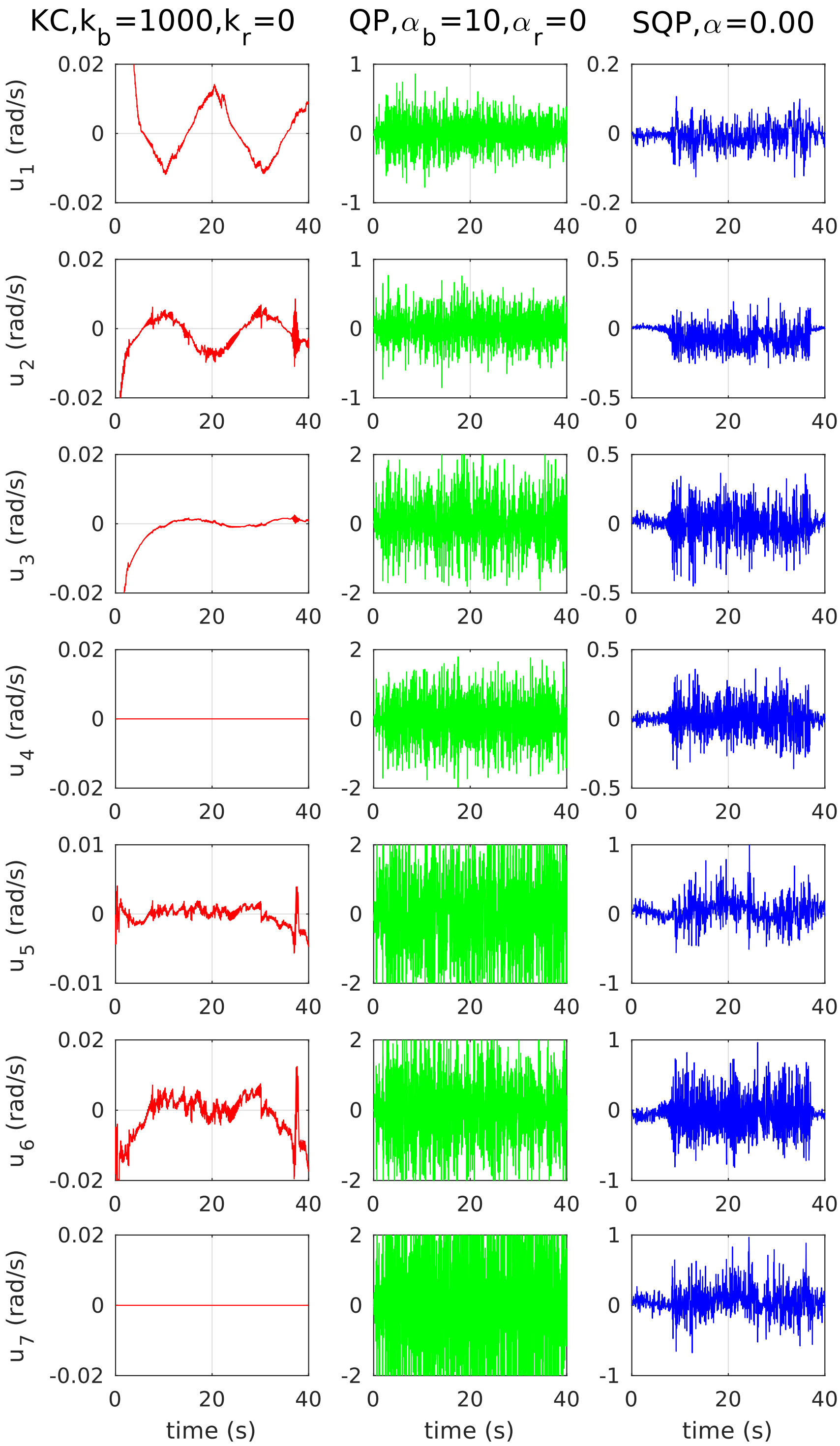


Figure 4.19: Joint control signals, part 1 of 2, manipulator satisfy a scleronomic constraint in simulation.

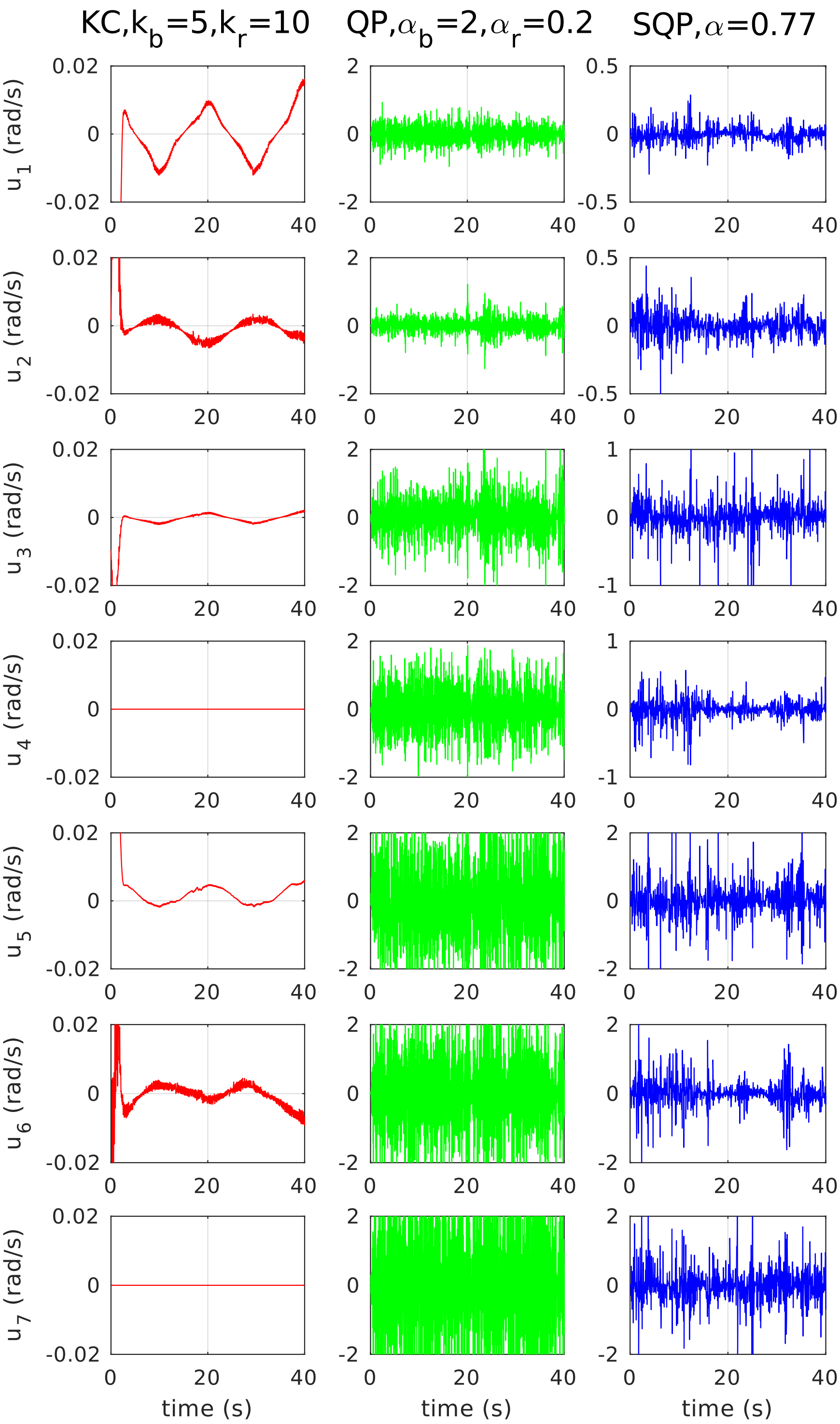


Figure 4.20: Joint control signals, part 2 of 2, manipulator satisfy a scleronomic constraint in simulation.

4.5 Experiment with a Scleronomic Constraint

In this experiment the Baxter has to track again the desired trajectory defined by (4.9) while satisfying a scleronomic constraint with $v_d(t) = 0$ and maximize the manipulability, the initial conditions for joint angles and end-effector position are again given by Tables 4.2 and Table 4.3, respectively. Only two methods, kinematic control and SQP, were able to reach solutions that satisfy the scleronomic constraint. The QP method was not able to ensure a satisfactory solution that satisfy the scleronomic constraint even with multiple attempts for different values of α_b, α_r and k_1. The set of solutions for kinematic control and SQP are represented in Figure. 4.21.

For the kinematic control in Figure 4.21, 38 solutions are defined with different values of k_b and k_r. Increasing only k_b from $k_b = 0$ to $k_b = 1000$ for a constant $k_b = 0$ leads to an increase of W_b: the values go from $W_b = 5.6 \times 10^{-3}$ to $W_b = 6.5 \times 10^{-3}$ while W_r holds in a value about $W_r = 2.45$. When increasing k_r from $k_r = 0$ to $k_r = 20$ (solution near $k_b = 5$,$k_r = 10$) for a constant $k_b = 0$ the values of W_r go from $W_r = 2.45$ to $W_r = 3.0$, also W_b increases to $W_b = 5.6 \times 10^{-3}$ to $W_b = 6.0 \times 10^{-3}$. This means there are some cooperative level between W_r and W_b, i.e., when an index increases the other increases too. Among the 38 solutions only 6 form the Pareto set. These 6 solutions have low and high values of k_b and k_r ($k_b = 5; k_r = 10$ and $k_b = 1000; k_r = 0$) as also intermediate values k_p and k_r ($k_b = 10; k_r = 5$ and $k_b = 500; k_r = 5$). For the values $k_b > 1000$ and $k_r > 10$ the system presents a huge increase error trajectory or the velocity in the constraint, and then solutions with these values are discarded.

Using the SQP method, a set of solutions is generated for $\alpha = \begin{bmatrix} 0.00 & 0.01 & \cdots & 0.99 & 1.00 \end{bmatrix}$. In this case one solution is a pair W_b and W_r for a fixed α, that way this set has 101 solution. For the SQP in Figure 4.21, only 2 solutions among 101 form the Pareto. As expected from (3.83) solutions with a high α value reach the best values of W_r, also some of these solutions reach the best values of W_b too (for example $\alpha = 0.90$) while others have low values of W_b. In another way, solutions with a low α value are clustered with a high W_b value and a low W_r value.

The trajectory error and the velocity in the constraint for kinematic control and SQP are represented in Figures 4.22 and 4.23, respectively. Only solutions belonging to Pareto set are presented, two from the kinematic control ($k_b = 5; k_r = 10$ and $k_b = 500; k_r = 5$) and two from the SQP ($\alpha = 0.89$ and $\alpha = 0.90$).

Regarding the trajectory error in Figure 4.22, by inspection all graphics seem to a have similar results, with the error in z axis, in general, being the more error prone. This is confirmed by Table 4.8 where in all graphics have similar integral values

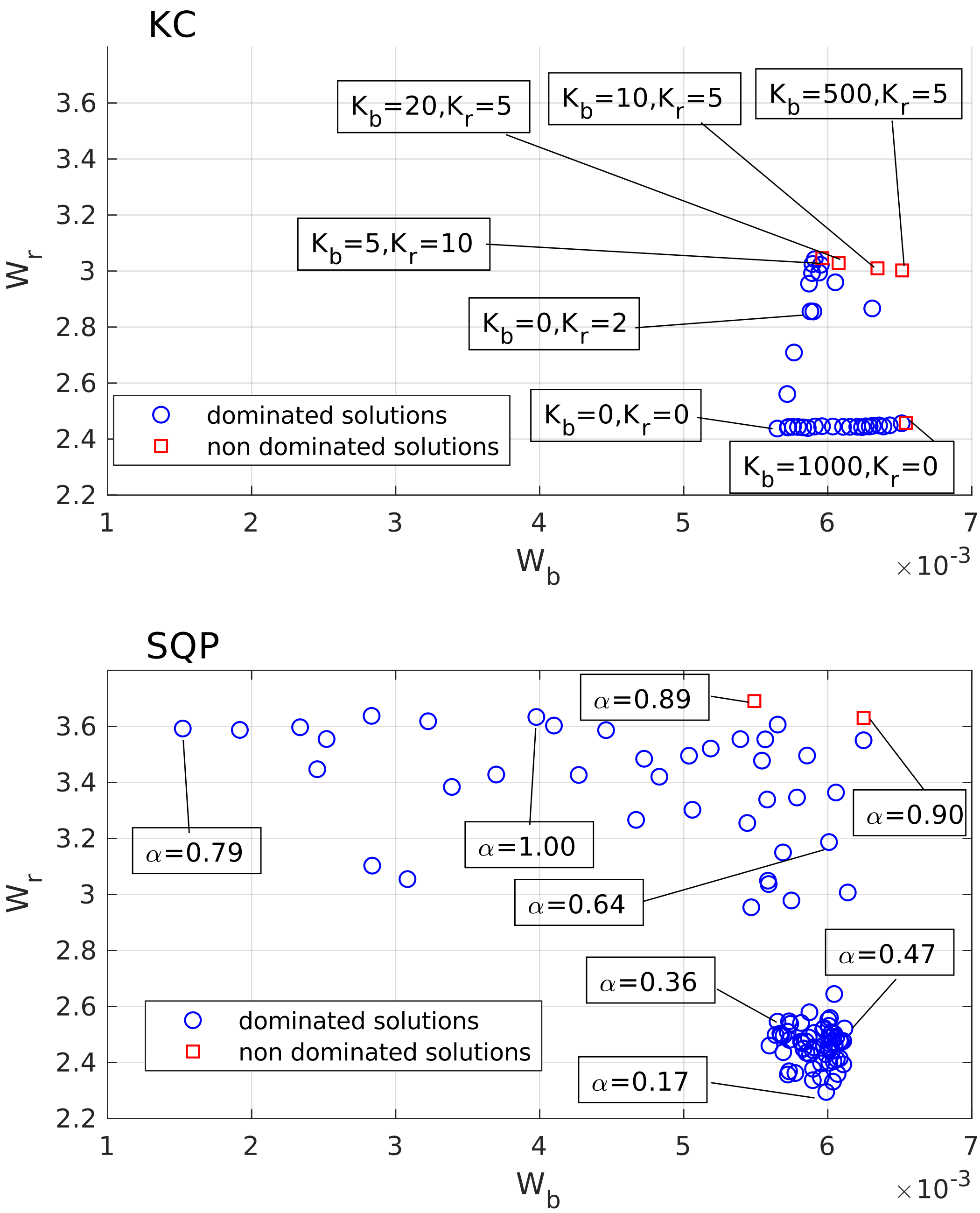


Figure 4.21: W_b and W_r, manipulator satisfy a scleronomic constraint in experiment.

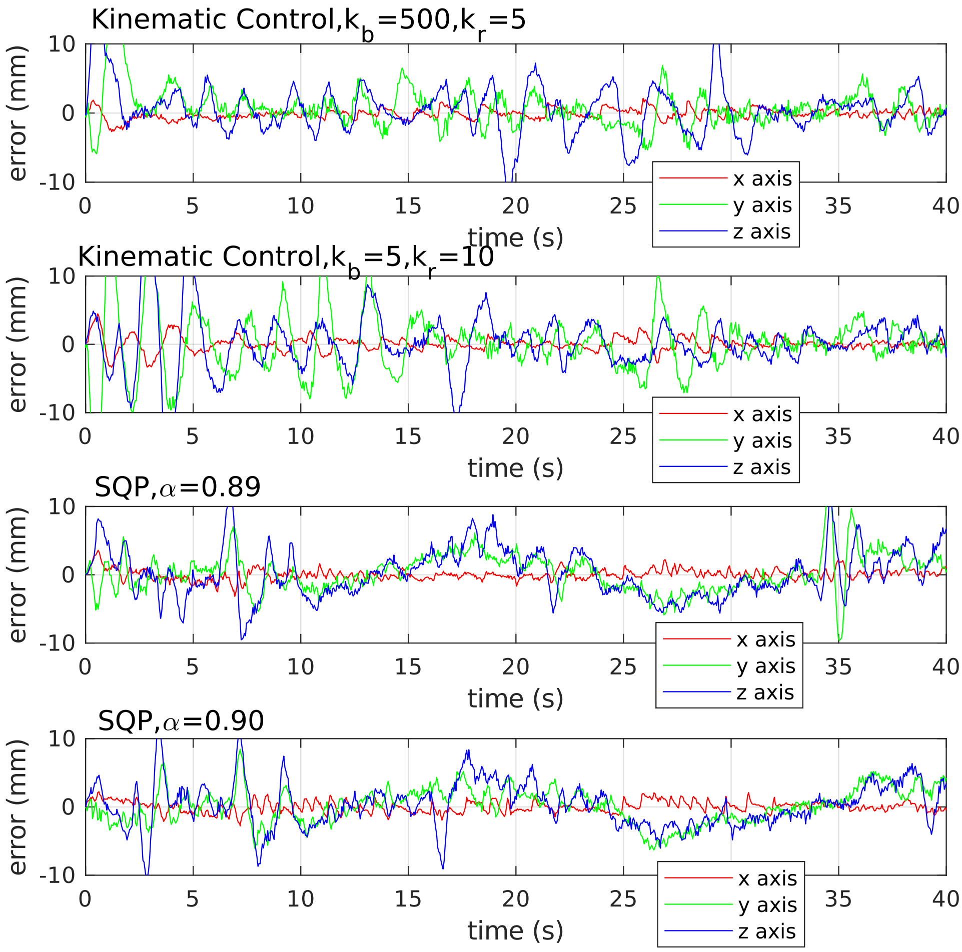


Figure 4.22: Trajectory error, manipulator satisfy a scleronomic constraint in experiment.

Table 4.8: Performance indexes, manipulator satisfy a scleronomic constraint in experiment.

Index	KC $k_b = 500$ $k_r = 5$	KC $k_b = 5$ $k_r = 10$	SQP $\alpha = 0.89$	SQP $\alpha = 0.90$
ISE e_x	1.05e-05	4.02e-05	2.37e-05	2.57e-05
ISE e_y	2.29e-04	9.49e-04	3.21e-04	2.64e-04
ISE e_z	1.33e-03	7.68e-04	5.16e-04	4.34e-04
IAE e_x	1.51e-02	2.80e-02	2.25e-02	2.58e-02
IAE e_y	7.11e-02	1.33e-01	8.74e-02	8.25e-02
IAE e_z	1.62e-01	1.16e-01	1.15e-01	1.02e-01
ITAE e_x	2.38e-01	4.20e-01	3.97e-01	4.54e-01
ITAE e_y	1.13e+00	1.82e+00	1.95e+00	1.73e+00
ITAE e_z	3.22e+00	1.64e+00	2.28e+00	1.93e+00
ITSE e_x	1.20e-04	4.25e-04	3.46e-04	4.13e-04
ITSE e_y	2.55e-03	8.15e-03	8.01e-03	5.62e-03
ITSE e_z	2.48e-02	6.69e-03	9.69e-03	7.09e-03
$\parallel e_x \parallel$	1.45e-02	2.84e-02	2.13e-02	2.27e-02
$\parallel e_y \parallel$	6.76e-02	1.38e-01	8.07e-02	7.29e-02
$\parallel e_z \parallel$	1.63e-01	1.24e-01	1.02e-01	9.37e-02

except some inferior performance solutions of kinematic control with $k_b = 5, k_r = 10$ in y (IAE and l_2 *norm*) and with $k_b = 500, k_r = 5$ in z (ISE and ITSE).

In Figure 4.23 it can be noted that v_c reaches the objective in all graphics, regardless the noise, except from the beginning until about 5 seconds. It is possible to note that the SQP solutions present less variance than kinematic control solutions.

In Figure 4.24 are presented the manipulability indexes behavior through time. In kinematic control graphics $w_b(\theta_{1,4}) \geq 0.15$ almost all time and $0.5 < w_r(\theta_{5,7}) < 0.1$, which is enough to keep the manipulator far from a singularity. In SQP for $w_b(\theta_{1,4})$, with $\alpha = 0.89$ lowers the value to near 0.1×10^{-3} but then increases to about 0.15×10^{-3} while with $\alpha = 0.90$ the value always remains close to 0.15. Still, in SQP $w_r(\theta_{5,7})$ has a similar evolution in the two graphics, increases at almost 0.1 e remains close to this value until the trajectory ends.

In Figures 4.25 and 4.26 are presented the control signals. The magnitude values are similar to both methods, kinematic control and SQP, with almost all values less than 0.1. Also, both methods present some peak values in the beginning of the trajectory. The signal variations are more abrupt in SQP than kinematic control.

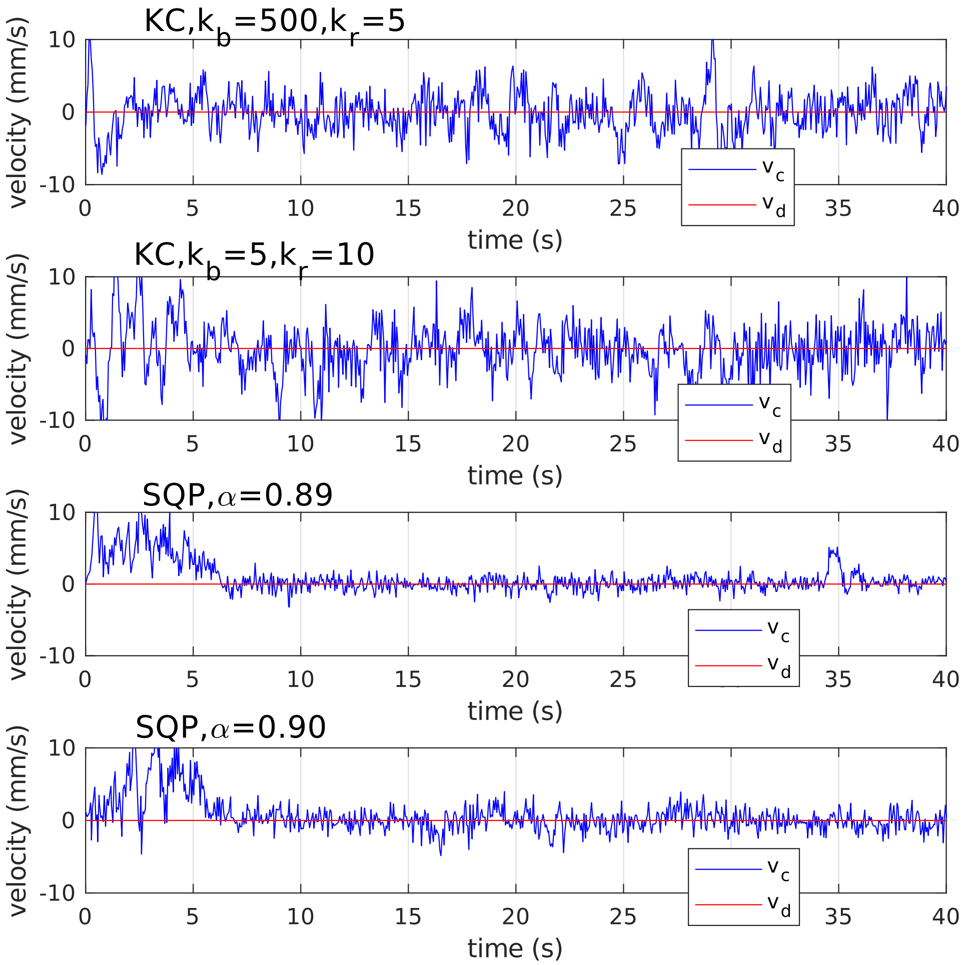


Figure 4.23: Velocity in the constraint, manipulator satisfy a scleronomic constraint in experiment.

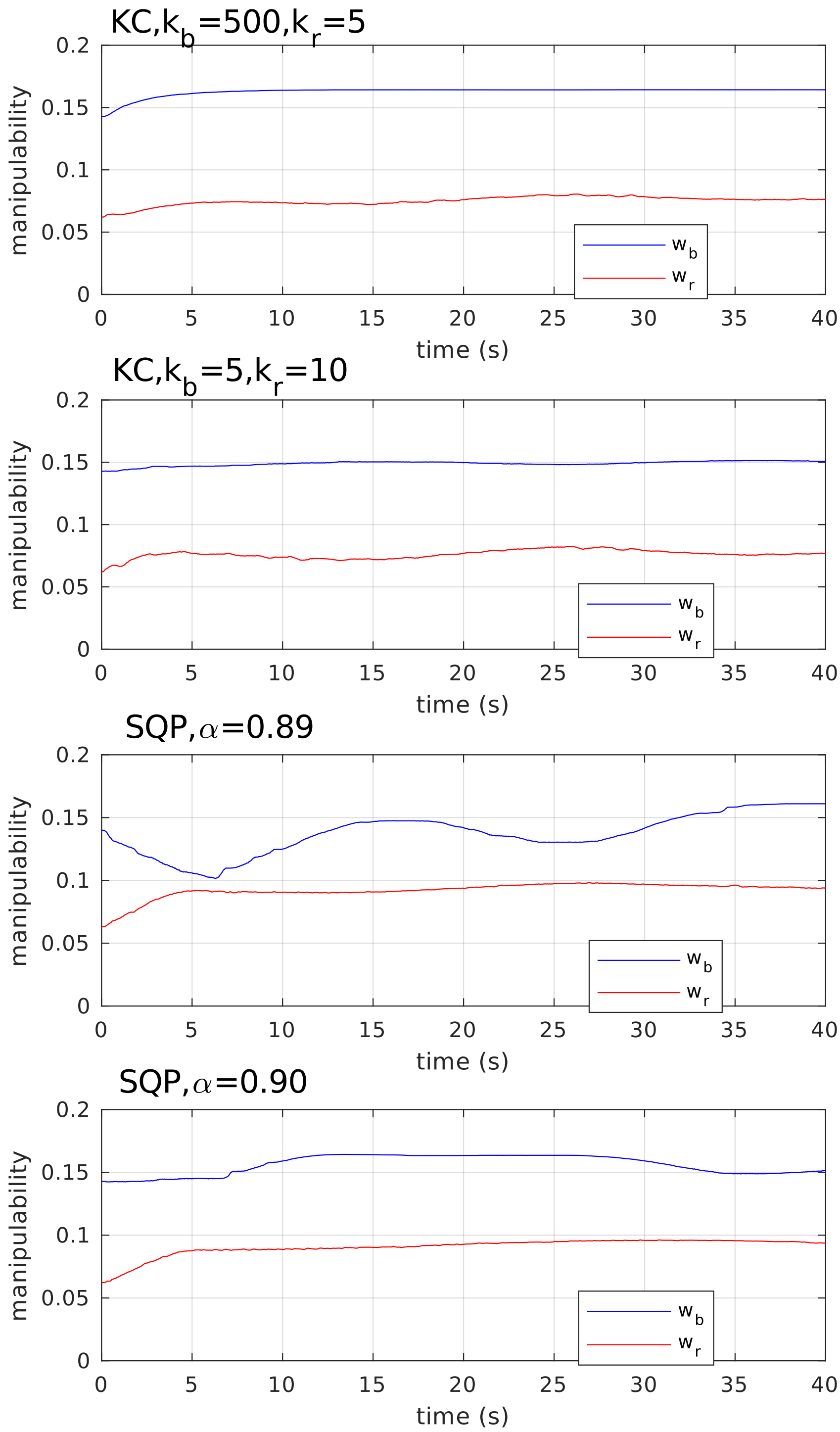


Figure 4.24: Manipulability indexes w_b and w_r (w_b is multiplied by 10^3), manipulator satisfy a scleronomic constraint in experiment.

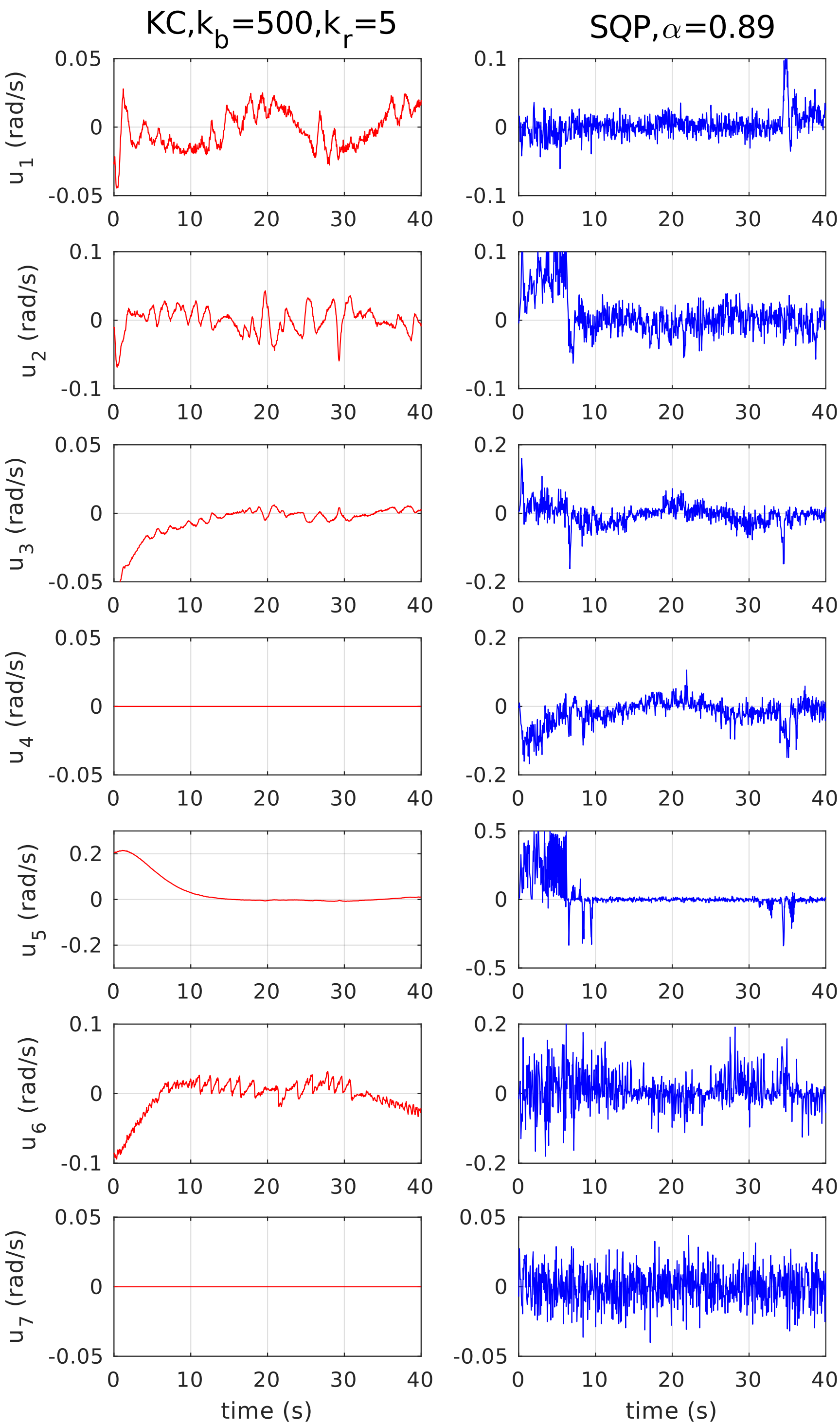


Figure 4.25: Joint control signals, part 1 of 2, manipulator satisfy a scleronomic constraint in experiment.

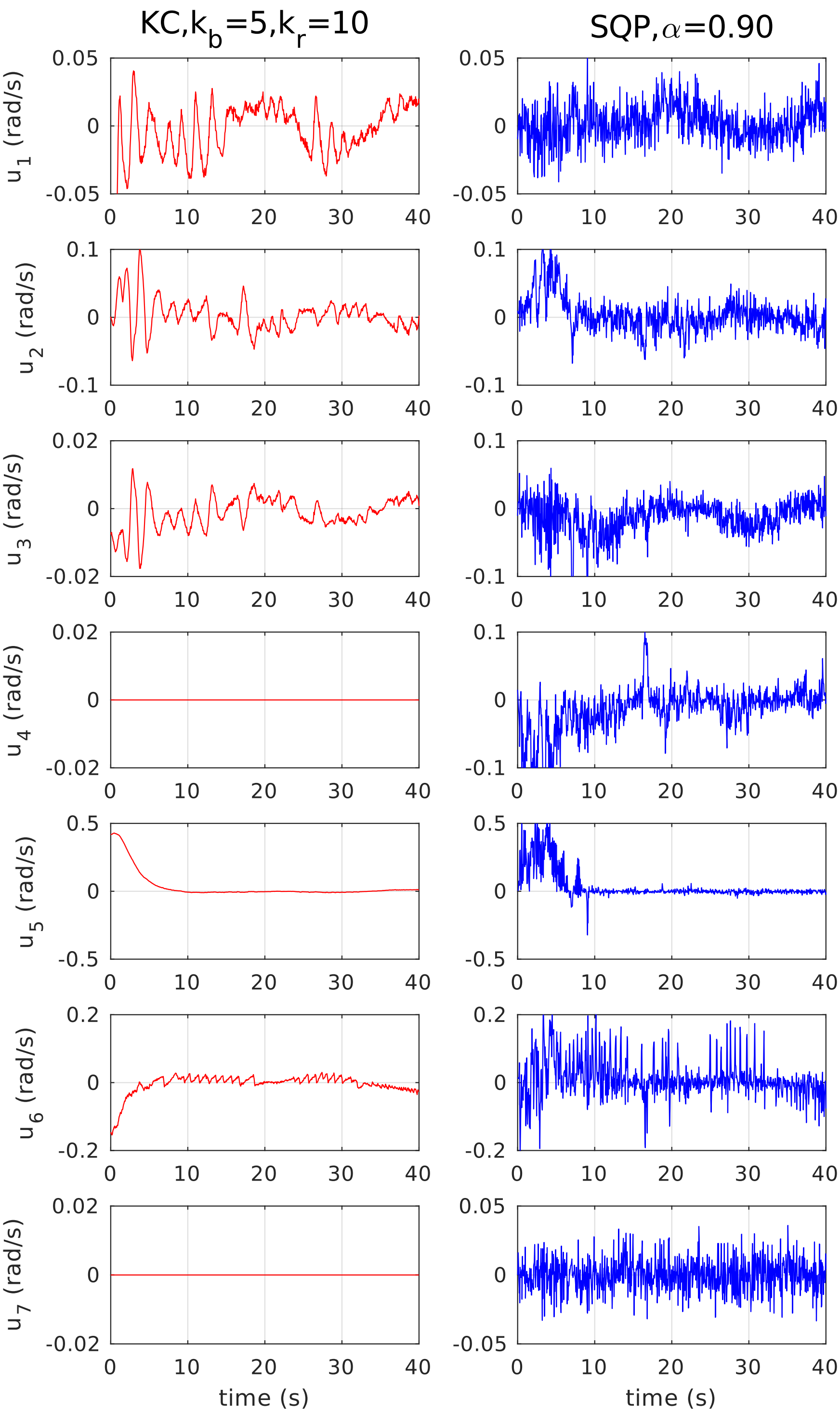


Figure 4.26: Joint control signals, part 2 of 2, manipulator satisfy a scleronomic constraint in experiment.

Table 4.9: Initial state of the joint angles for the desired trajectory defined in (4.15).

Joint angle	Value (rad/s)
θ_1	0
θ_2	$-\pi/6$
θ_3	$\pi/2$
θ_4	$\pi/4$
θ_5	$-\pi/3$
θ_6	$\pi/4$
θ_7	0

Table 4.10: Initial end-effector position for the desired trajectory defined in (4.15).

Axis	Value (mm)
p_x	961
p_y	-438
p_z	593

4.6 Experiment with a Rheonomic Constraint

In this experiment the manipulator is subject to a rheonomic constraint with $v_d(t) = 0.01\sin(t)m/s$ while end-effector tracks the following desired trajectory:

$$p_d(t) = \begin{bmatrix} p_x(0) + 15\sin(\pi t/20) \\ p_y(0) + 66\cos(2\pi t/20) - 66 \\ p_z(0) + 30\sin(2\pi t/20) \end{bmatrix} mm, \tag{4.15}$$

where the initial state of the joint angles and the initial end-effector position are defined in Table 4.9 and Table 4.10, respectively. The task execution time is 12 s. The desired trajectory in (4.15) considering the values given by Tables 4.9 and 4.10 is visualized in Figure 4.27. Only the SQP method achieve valid solutions for the rheonomic constraints.

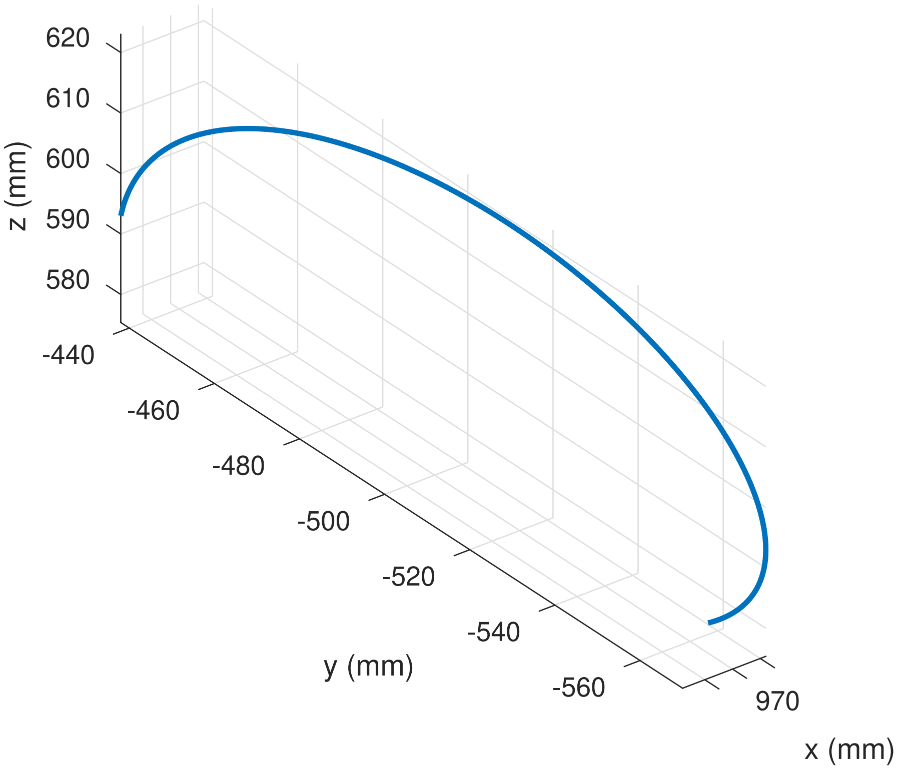


Figure 4.27: Desired trajectory defined in (4.15).

Again using the SQP method a set of solutions is generated for $\alpha = \begin{bmatrix} 0 & 0.01 & \cdots & 0.99 & 1 \end{bmatrix}$. The set of solutions is represented in Figure 4.28 and six solutions form the Pareto set. It is worth mentioning that not all solutions reach feasible values for trajectory error or velocity in the constraint. So only 41 solutions are represented in Figure 4.28 all of them with $\alpha \leq 0.45$ and all Pareto solutions have $\alpha \geq 0.28$. In this way, very low α values can not reach the best values for W_b or W_r, at least most of them are feasible.

The trajectory error and the velocity in the constraint for two α values ($\alpha = 0.33$ and $\alpha = 0.36$) are represented in Figures 4.29 and 4.30, respectively. The trajectory error are most time between -5 mm and 5 mm, except for the trajectory end with $\alpha = 0.28$. The Table 4.11 shows the performance index for the trajectory error, the indexes for $\alpha = 0.33$ are slightly better than indexes for $\alpha = 0.36$, except for ISE, IAE and l_2 *norm* in z axis.

The velocity in the constraint, Figure 4.30, had a hard time to follow $v_d(t)$, being out of phase. But al least manage to satisfy at some level the constraint.

The Figure 4.31 shows the manipulability indexes. In the two graphics $w_r(\theta_{5,7})$ is almost the same with a value slightly above 0.05. In the case of $w_b(\theta_{1,4})$, the graphic with $\alpha = 0.33$ has a steeper slope reaching at the end a higher value in relation to $\alpha = 0.36$.

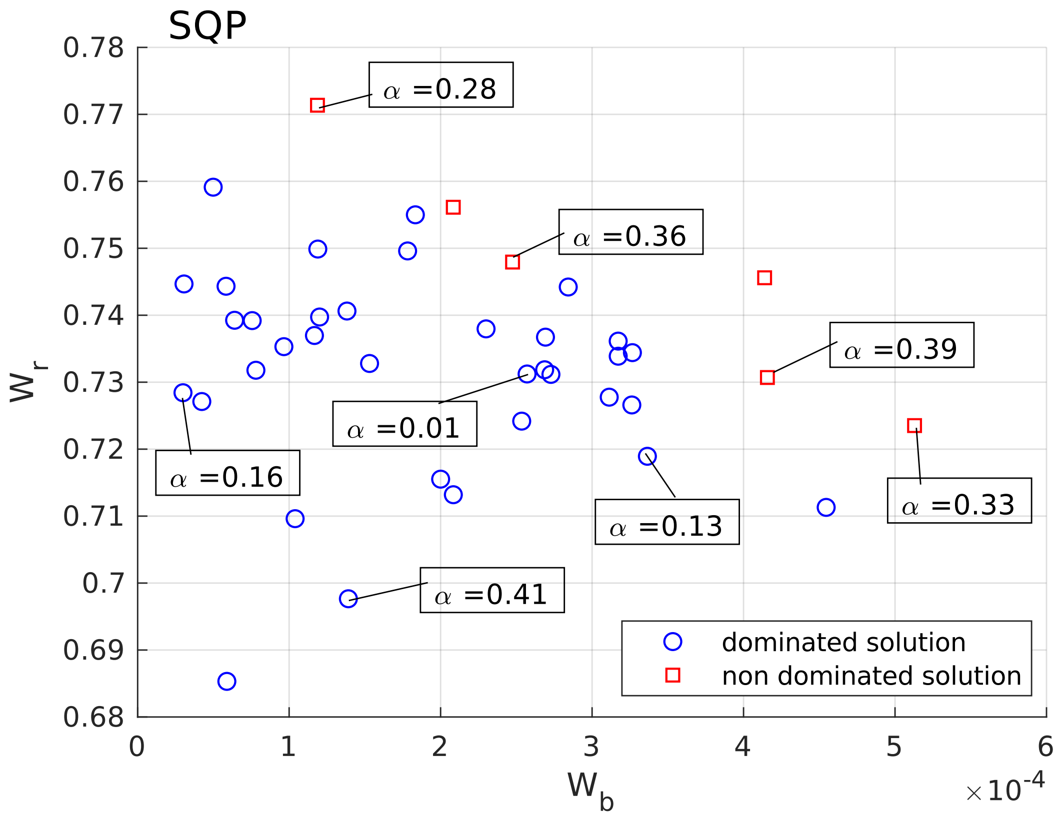


Figure 4.28: W_b and W_r, manipulator satisfy a rheonomic constraint in experiment.

Table 4.11: Performance indexes, manipulator satisfy a rheonomic constraint in experiment.

Index	SQP $\alpha = 0.33$	SQP $\alpha = 0.36$
ISE e_x	4.15e-06	5.93e-06
ISE e_y	5.72e-05	6.23e-05
ISE e_z	6.70e-05	6.43e-05
IAE e_x	5.85e-03	7.07e-03
IAE e_y	1.99e-02	2.14e-02
IAE e_z	2.49e-02	2.31e-02
ITAE e_x	4.02e-02	5.15e-02
ITAE e_y	1.27e-01	1.45e-01
ITAE e_z	1.50e-01	1.57e-01
ITSE e_x	3.06e-05	4.86e-05
ITSE e_y	3.51e-04	4.32e-04
ITSE e_z	3.99e-04	4.70e-04
$\parallel e_x \parallel$	9.12e-03	1.09e-02
$\parallel e_y \parallel$	3.38e-02	3.53e-02
$\parallel e_z \parallel$	3.66e-02	3.59e-02

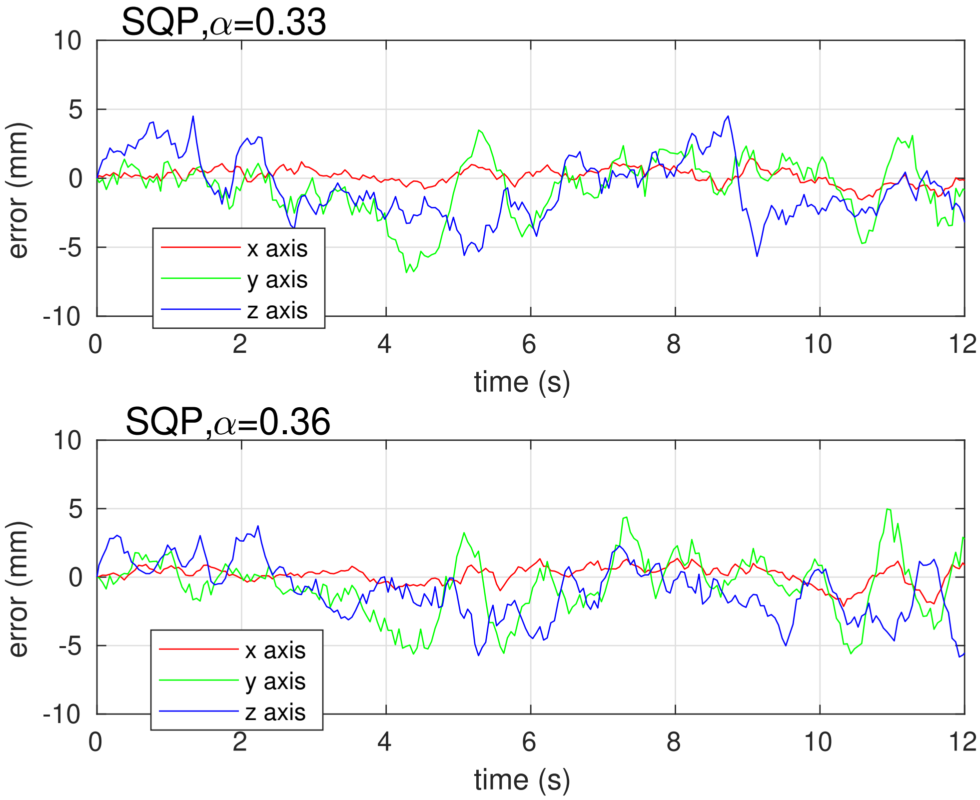


Figure 4.29: Trajectory error, manipulator satisfy a rheonomic constraint in experiment.

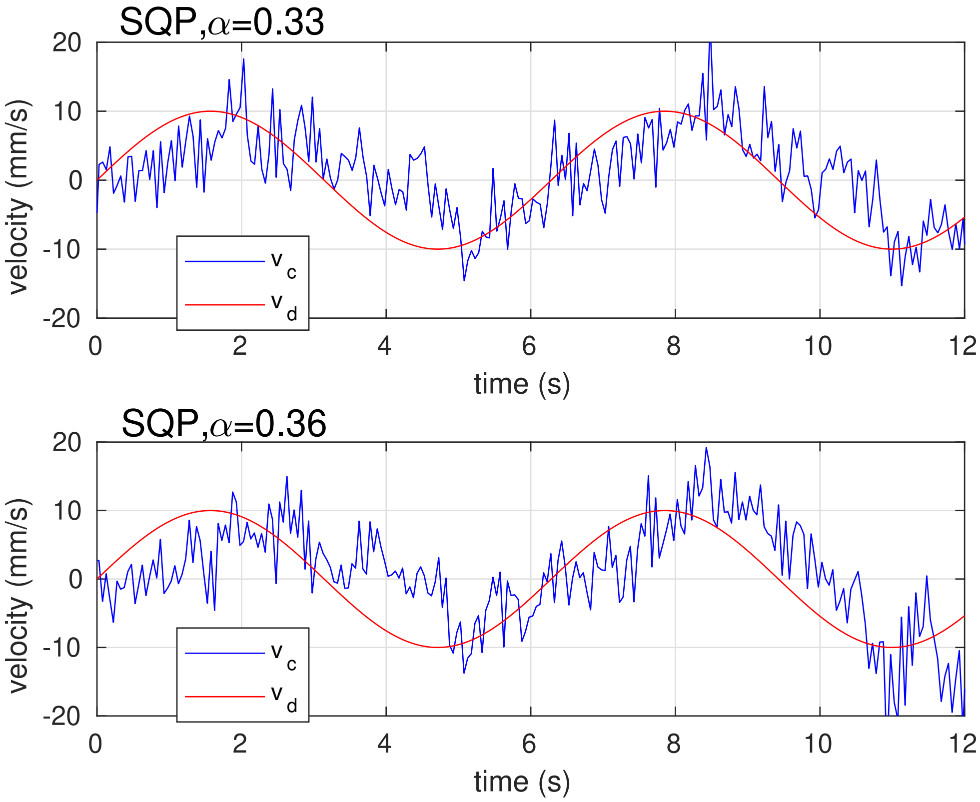


Figure 4.30: Velocity in the constraint, manipulator satisfy a rheonomic constraint in experiment.

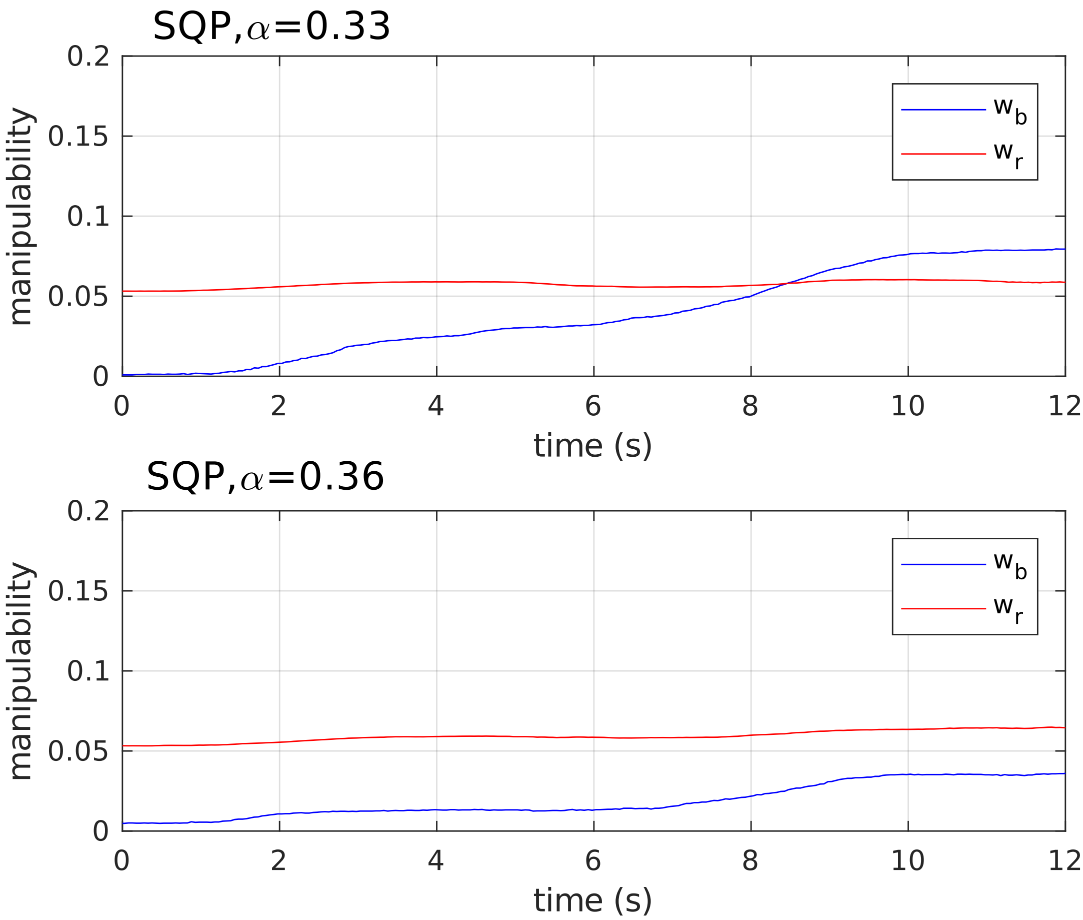


Figure 4.31: Manipulability indexes w_b and w_r (w_b is multiplied by 10^3), manipulator satisfy a rheonomic constraint in experiment.

Finally, the Figure 4.32 shows the joint control signals. The two graphics are very similar and it is noted that the control signal for the first joint is a sinusoidal signal with the same frequency that the desired velocity of the rheonomic constraint.

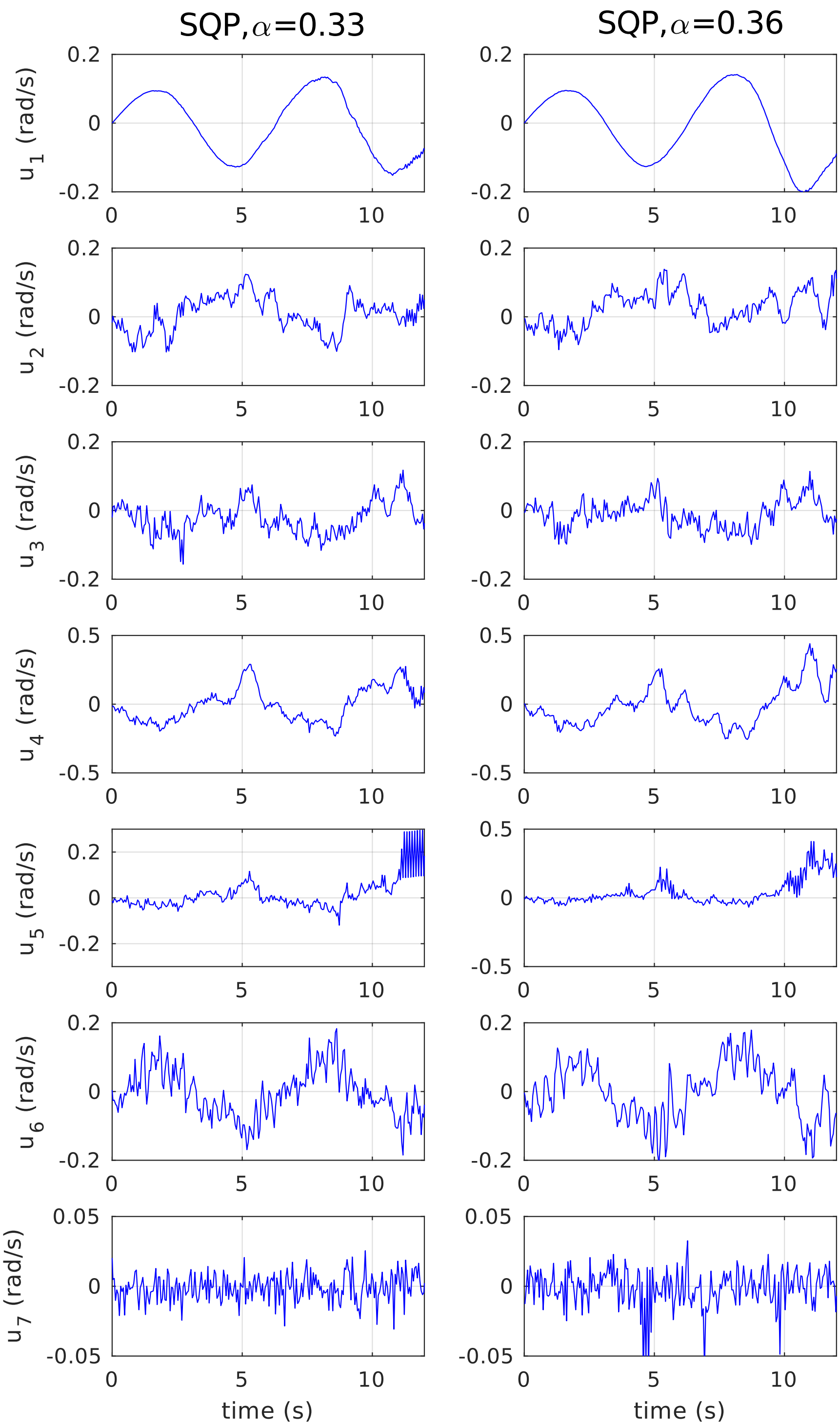


Figure 4.32: Joint control signals, manipulator satisfy a rheonomic constraint in experiment.

Chapter 5

Conclusions

In this thesis the following problem is presented: the end effector of a serial redundant manipulator has to track a desired trajectory while a point in the kinematic chain satisfy a holonomic constraint and one or more manipulability indexes are maximized. Three methods are discussed in order to solve the problem: kinematic control, quadratic programming and sequential quadratic programming.

Formulate the trajectory tracking problem using sequential quadratic algorithm method is an alternative solution to kinematic control. First advantage is the ease of integrating constraints in the optimization formulation, these constraints can be considered straightforward without any linearization or curve fitting, there is also scalability of defining multiple constraints without defining any constrained Jacobian matrix. Second advantage is the possibility to maximize any objective function without lose its shape for a curve fitting and with the weight sum approach easily transforms the problem in a multi objective one. The third advantage is there no need for derivative of trajectory track.

The formulation of the tracking problem through SQP with respect to QP also has advantages. The first is the lack of linearization of the objective function. The second is the incorporation of the prediction trajectory error through forward kinematics, non existent in the formulation by QP because it is a nonlinear function. The third is the same advantage in relation to kinematic control, the absence of derivative of trajectory tracking.

In the experiments the SQP is the only method that is able to satisfy both the scleronomic and rheonomic constraint. As the Baxter is a low position accuracy robot the QP method have difficult to track the trajectory because its use the end effector velocity (which is more inaccurate) to adjust the trajectory, so it does not have success in any experiment.

In the scleronomic experiment although the kinematic control reached higher values for W_b (with the SQP slightly behind), the values of W_r are much lower than in SQP. In the rheonomic experiment the kinematic control is unable to guarantee that the velocity constraint is satisfied. So, considering only the experiments in this thesis the SQP had a better performance than the kinematic control. Further study is needed to verify why kinematic control failed in the rheonomic experiment.

Some topics for future improvement utilizing the SQP method are listed:

- Add negative feedback information about the velocity in the constraint, so the method could improve a holonomic constraint that is not being satisfied in a system application level although it may be satisfied at the algorithm level.
- Study the SQP stability, which need assumptions and conditions e needs study before the method can be applied in critical systems. extending the results of actual literature.
- Study how complexity time of SQP applied in the trajectory track behaves in order to have an estimate of the convergence time
- Apply parallelism on SQP to improve the convergence time. In [32] is introduced a way to accelerate the SQP by minimizing functions evaluation on a graphical processors unit.
- Integrate the trajectory tracking method via SQP using the dynamic equations of manipulators.

Bibliography

[1] The cuter/st test problem set. *http://www.cuter.rl.ac.uk/Problems/mastsif.shtml*, Accessed: 01/07/2019.

[2] T. Ando. Majorization, doubly stochastic matrices, and comparison of eigenvalues. *Linear Algebra and its Applications*, 118:163–248, 1989.

[3] J. Arora. *Introduction to Optimum Design*. Academic Press, 2012.

[4] M. Azad, J. Babič, and M. Mistry. Dynamic manipulability of the center of mass: A tool to study, analyse and measure physical ability of robots. *IEEE International Conference on Robotics and Automation*, pages 3484–3490, 2017.

[5] M. Bartholomew-Biggs. *Nonlinear Optimization with Engineering Applications*. Springer, 2008.

[6] I. Bomze, R. Fletcher, V. Demyanov, and T. Terlaky. *Nonlinear Optimization*. Springer, 2007.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009.

[8] R. Campa and H. Torre. Pose control of robot manipulators using different orientation representations: A comparative review. *American Control Conference*, pages 2855–2860, 2009.

[9] L. Capisani and A. Ferrara. Trajectory planning and second-order sliding mode motion/interaction control for robot manipulators in unknown environments. *IEEE Transactions on Industrial Electronics*, 59(8):3189–3198, 2012.

[10] F. Cardoso and F. Lizarralde. Multiobjective manipulability in trajectory tracking for constrained redundant robot manipulators. *Simpósio Brasileiro de Automação Inteligente*, pages 1773–1778, 2017.

[11] F. Cardoso and F. Lizarralde. Comparison of methods for trajectory tracking for redundant manipulators under holonomic scleronomic constraints. *Congresso Brasileiro de Automática*, 2018.

[12] D. Carrasco and M. Salgado. Optimal multivariable controller design using an itse performance index. *International Journal of Control*, 83(11):2340–2353, 2010.

[13] J. Castro. Minimum-distance controlled perturbation methods for large-scale tabular data protection. *European Journal of Operational Research*, 171(1):39–52, 2006.

[14] D. Chen and R. Plemmons. Nonnegativity constraints in numerical analysis. *The Birth of Numerical Analysis*, pages 109–139, 2009.

[15] S. Chiaverini, G. Oriolo, and I. Walker. *Kinematically Redundant Manipulators.* Handbook of Robotics. Springer, 2008.

[16] F. Coutinho. *Controle de Manipulador Redundante com Restrições Cinemáticas Aplicado a Cirurgias Robóticas Assistidas.* Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, MSc dissertation, Rio de Janeiro, Brazil, 2015, In Portuguese.

[17] F. Coutinho, C. Pham, P. From, and F. Lizarralde. Abordagem analítica para controle no espaço operacional de manipuladores com restrições cinemáticas. *XX Congresso Brasileiro de Automática*, pages 958–964, 2014, In Portuguese.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[19] R. Dembo and U. Tulowitzki. *On the minimization of quadratic functions subject to box constraints.* Working Paper 71, Yale University, 1983.

[20] R. Dorf and R. Bishop. *Modern Control Systems.* Pearson, 2016.

[21] K. Dufour and W. Suleiman. On integrating manipulability index into inverse kinematics solver. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6967–6972, 2017.

[22] J. Everist and W. Shen. Mapping opaque and confined environments using proprioception. *IEEE International Conference on Robotics and Automation*, pages 1041–1046, 2009.

[23] T. Feng, Y. Kobayashi, M. Minami, and A. Yanou. Dynamic reconfiguration manipulability analysis of redundant robot. *IEEE International Conference on Mechatronics and Automation*, pages 51–56, 2013.

[24] J. Fiala and B. Marteau. Nonlinear optimization: A comparison of two competing approaches, active-set sqp vs. ipm. *https://www.nag.co.uk/market/nonlinear-optimization-comparison.pdf*, Accessed in 01/07/2019.

[25] P. From. On the kinematics of robotic-assisted minimally invasive surgery. *Modeling, Identification and Control*, 34:69–82, 2013.

[26] P. From, A. Robertsson, and R. Johansson. On the manipulability of velocity-constrained serial robotic manipulators. *In World Congress*, 19(1):10934–10939, 2014.

[27] M. Galicki. Real-time constrained trajectory generation of mobile manipulators. *Robotics and Autonomous Systems*, 78(1):49–62, 2016.

[28] P. Gill, W. Murray, M. Saunders, and M. Wright. *Constrained Nonlinear Programming*. Handbooks in Operations Research and Management Science, Optimization, Volume 1. Elsevier, 1989.

[29] D. Glynn. The permanent of a square matrix. *European Journal of Combinatorics*, 31(7):1887–1891, 2010.

[30] N. Gould, D. Orban, and P. Toint. Cutest: A constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015.

[31] L. Hosford. *Development and Testing of an Impedance Controller on an Anthropomorphic Robot for Extreme Environment Operations*. Department of Mechanical Engineering, Massachusetts Institute of Technology, MSc dissertation, Massachusetts, USA, 2016.

[32] X. Hu, C. Douglas, R. Lumley, and M. Seo. Gpu accelerated sequential quadratic programming. *International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, pages 3–6, 2017.

[33] Y. Huang, X. Zhang, X. Chen, and J. Ota. Vision-guided peg-in-hole assembly by baxter robot. *Advances in Mechanical Engineering*, 9(12):1–9, 2017.

[34] R. Jazar. *Theory of Applied Robotics: Kinematic, Dynamics, and Control*. Springer, 2010.

[35] R. Jazar. *Advanced Dynamics: Rigid Body, Multibody and Aerospace Applications*. Wiley, 2011.

[36] P. Jimack. *Parallel optimization for a finite element problem from nonlinear elasticity*. School of Computer Studies, report 91.22, University of Leeds, 1991.

[37] L. Joseph. *Mastering ROS for Robotics Programming*. Packt Publishing, 2015.

[38] Z. Ju, C. Yang, and H. Ma. Kinematics modeling and experimental verification of baxter robot. *Chinese Control Conference*, pages 8518–8523, 2014.

[39] M. Khadem, L. Cruz, and C. Bergeles. Force/velocity manipulability analysis for 3d continuum robots. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4920–4926, 2018.

[40] J. Kim and P. Khosla. Dexterity measures for design and control of manipulators. *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 758–763, 1991.

[41] D. Kraft. A software package for sequential quadratic programming. *Deutsche Forschungs- und Versuchhsanstalf fur Luft- un Raumfahrt*, pages 1–33, 1988.

[42] S. Lee. Sequential quadratic programming based global path re-planner for a mobile manipulator. *International Journal of Control, Automation, and Systems*, 4(3):318–324, 2006.

[43] Z. Li, B. Liao, F. Xu, and D. Guo. A new repetitive motion planning scheme with noise suppression capability for redundant robot manipulators. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pages 1–11, 2018, Early Access.

[44] X. Liang, Y. Wan, and C. Zhang. Task space trajectory control of robot manipulators with uncertain kinematics and dynamics. *Mathematical Problems in Engineering*, pages 1–19, 2017.

[45] Y. Liao and C. Fan. Multi-objective motion planning of space flexible manipulator system. *IEEE International Conference on Real-time Computing and Robotics*, pages 477–482, 2016.

[46] P. Long and T. Padir. Evaluating robot manipulability in constrained environments by velocity polytope reduction. *IEEE-RAS International Conference on Humanoid Robots*, pages 497–502, 2018.

[47] M. Lorenz, J. Brinker, I. Prause, and B. Corves. Power manipulability analysis of redundant actuated parallel kinematic manipulators with different types of actuators. *IEEE International Conference on Robotics and Automation*, pages 2129–2136, 2016.

[48] S. Lyle and N. Nichols. *Numerical Methods for Optimal Control Problems with State Constraints*. Numerical Analysis Report 8/91, Dept of Mathematics, University of Reading, 1991.

[49] K. Madsen and H. Schjaer-Jacobsen. Linearly constrained minimax optimization. *Mathematical Programming*, 14:208–223, 1978.

[50] A. Martinez and E. Fernándes. *Learning ROS for Robotics Programming*. Packt Publishing, 2013.

[51] D. Menell, C. Lin, R. Chitnis, S. Russell, and P. Abbeel. Sequential quadratic programming for task plan optimization. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5040–5047, 2016.

[52] H. Minc. Theory of permanents. *Linear Algebra and its Applications*, 21:109–148, 1987.

[53] D. Mori and G. Ishigami. Generalized force-and-energy manipulability for design and control of redundant robotic arm. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1131–1136, 2015.

[54] R. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[55] N. Naksuk and C. Lee. Zero moment point manipulability ellipsoid. *IEEE International Conference on Robotics and Automation*, pages 1970–1975, 2006.

[56] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2006.

[57] J. O'Kane. *A Gentle Introduction to ROS*. http://www.cse.sc.edu/ jokane/agitr/, 2013, Accessed in 01/07/2019.

[58] J. Papastavridis. *Analytical Mechanics: A Comprehensive Treatise on the Dynamics of Constrained Systems; for Engineers, Physicists, and Mathematicians*. Oxford University Press, 2002.

[59] D. Pardo, Lukas L. Moller, M. Neunert, A. Winkler, and J. Buchli. Evaluating direct transcription and nonlinear optimization methods for robot motion planning. *IEEE Robotics and Automation Letters*, 1(2):946–953, 2016.

[60] J. Pei and J. Cheng. Optimization of force directional manipulability of dexterous robot hand. *International Conference on System Science, Engineering Design and Manufacturing Informatization*, pages 226–229, 2010.

[61] R. Perez, P. Jansen, and J. Martins. pyopt: A python-based object-oriented framework for nonlinear constrained optimization. *Structures and Multidisciplinary Optimization*, 45(1):101–118, 2012.

[62] C. Pham, F. Coutinho, A. Leite, F. Lizarralde, P. From, and R. Johansson. Analysis of a moving remote center of motion for robotics-assisted minimally invasive surgery. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1440–1446, 2015.

[63] C. Pham, F. Coutinho, F. Lizarralde, L. Hsu, and P. From. An analytical approach to operational space control of robotic manipulators with kinematic constraints. *IFAC In World Congress*, 19(1):8509–8515, 2014.

[64] G. Pratt and M. Williamson. Series elastic actuators. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 399–406, 1995.

[65] E. Ray. *Learning XML.* O'Reilly, 2001.

[66] R. Roberts, H. Yu, and A. Maciejewski. Characterizing optimally fault-tolerant manipulators based on relative manipulability indices. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3925–3930, 2007.

[67] M. Rosenstein and R. Grupen. Velocity-dependent dynamic manipulability. *IEEE International Conference on Robotics and Automation*, pages 2424–2429, 2002.

[68] L. Rozo, N. Jaquier, S. Calinon, and D.Caldwell. Learning manipulability ellipsoids for task compatibility in robot manipulation. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3183–3189, 2017.

[69] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics Modeling, Planning and Control.* Springer, 2009.

[70] E. Silva, F. Costa, E. Bicho, and W. Erlhagen. Nonlinear optimization for human-like movements of a high degree of freedom robotics arm-hand system. *Computational Science and Its Applications*, pages 327–342, 2011.

[71] J. Slotine and W. Li. *Applied Nonlinear Control.* Prentice-Hall, 1991.

[72] M. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2005.

[73] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets - optimization test problems. *https://www.sfu.ca/ ssurjano/optimization.html*, Accessed in 01/07/2019.

[74] P. Teodorescu. *Mechanical Systems, Classical Models. Volume 1 Particle Mechanics*. Springer, 2007.

[75] J. Thygeson, S. Moe, K. Pettersen, and J. Gravdahl. Kinematic singularity avoidance for robot manipulators using set-based manipulability tasks. *IEEE Conference on Control Technology and Applications*, pages 142–149, 2017.

[76] A. Torabi, M. Khadem, K. Zareinia, G. Sutherland, and M. Tavakoli. Manipulability of teleoperated surgical robots with application in design of master/slave manipulators. *International Symposium on Medical Robotics*, pages 1–6, 2018.

[77] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann. Manipulability analysis. *IEEE-RAS International Conference on Humanoid Robots*, pages 568–573, 2012.

[78] L. Vandenberghe. The cvxopt linear and quadratic cone program solvers. *http://www.seas.ucla.edu/ vandenbe/publications/coneprog.pdf*, pages 1–30, Accessed in 01/07/2019.

[79] H. Wang. Towards manipulability of bilateral teleoperators with time-varying delay. *Chinese Automation Congress*, pages 946–951, 2018.

[80] T. Watanabe. Manipulability measures taking necessary joint torques for grasping into consideration. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 598–603, 2010.

[81] T. Yoshikawa. Manipulability of robot mechanisms. *The International Journal of Robotics Research*, 4(2):3–9, 1985.

[82] X. Yun and N. Sarkar. Unified formulation of robotic systems with holonomic and nonholonomic constraints. *IEEE Transactions on Robotics an Automation*, 14(4):640–650, 1998.

[83] T. Zhang, F. Yu, M. Minami, O. Yasukura, W. Song, A. Yanou, and M. Deng. Non-singular configuration analyses of redundant manipulators for optimizing avoidance manipulability. *International Conference on Soft Computing and Intelligent Systems*, pages 963–970, 2010.

[84] Y. Zhang and l. Jin. *Robot Manipulator Redundancy Resolution.* Wiley, 2018.

[85] Y. Zhang and Z. Zhang. *Repetitive Motion Planning and Control of Redundant Robot Manipulators.* Springer, 2013.

Appendix A

Proof of Theorems

A.1 Definitions

Considering the following nonlinear problem:

$$\min_{u} \quad f(u) \ \in \mathbb{R}, \tag{A.1a}$$

$$\text{subject to:} \quad c_i(u) = 0, \quad i \in \mathcal{E}; \tag{A.1b}$$

$$c_i(u) \geq 0, \quad i \in \mathcal{I}. \tag{A.1c}$$

Definition A.1.1. The active set $\mathcal{A}(u)$ at any feasible solution u in (A.1) consists of the equality constraints indexes from $\mathcal{E}$ together with the indexes of the inequalities constraints i for which $c_i(u) = 0$,

$$\mathcal{A}(u) = \mathcal{E} \cup \{i \in \mathcal{I} | c_i(u) = 0\}, \tag{A.2}$$

and at any feasible solution u, the inequality constraint $i \in \mathcal{I}$ is said to be active if $c_i(u) = 0$ and inactive if the strict inequality $c_i(u) > 0$ is satisfied.

Definition A.1.2. Given a solution u and the active set $\mathcal{A}(u)$ defined in Definition A.1.1, we say that the linear independence constraint qualification holds if the set of active constraint gradients $\{\nabla c_i(u), i \in \mathcal{A}(u)\}$ is linearly independent.

Definition A.1.3. Given a solution u in (A.1), considering the functions f and c_i in (A.1) are continuously differentiable, and the Definition A.1.2 holds at u. Then, there is a Lagrange multiplier vector λ, with components λ_i, $i \in \mathcal{E} \cup \mathcal{I}$, such the following conditions are satisfied at (u, λ):

$$\nabla \mathcal{L}(u, \lambda) = 0, \tag{A.3a}$$

$$c_i(u) = 0, \quad i \in \mathcal{E}, \tag{A.3b}$$

$$c_i(u) \geq 0, \quad i \in \mathcal{I}, \tag{A.3c}$$

$$\lambda_i \geq 0, \quad i \in \mathcal{I}, \tag{A.3d}$$

$$\lambda_i c_i(u) \geq 0, \quad i \in \mathcal{E} \cup \mathcal{I}, \tag{A.3e}$$

the conditions in (A.3) are the Karush-Kuhn-Tucker (KKT) conditions.

Definition A.1.4. Given a solution u in (A.1), a Lagrange multiplier vector λ satisfying the (KKT) conditions and $H(\mathcal{L}(u, \lambda))$ is symmetric and positive definite. Then, u is a strict local solution for (A.1), i.e., $f(u) < f(u^*)$ for all solutions u^* in the neighborhood of u.

A.2 Proof of Theorem 1

This proof is based in [56]. Considering the QP optimization problem given by:

$$\min_u \quad f(u) = \frac{1}{2} u^T C u + c^T u \in \mathbb{R}, \tag{A.4a}$$

$$\text{subject to:} \quad {B_i}^T u = s_i, \quad i \in \mathcal{E}; \tag{A.4b}$$

$${B_i}^T u < s_i, \quad i \in \mathcal{I}. \tag{A.4c}$$

Given a solution u_k in (A.4) that not minimizes the objective function, a search direction d_k is defined by:

$$d_k = u - u_k \tag{A.5}$$

Substituting (A.5) in (A.4a) results in:

$$f(u_k + d_k) = \frac{1}{2} u_k^T C u_k + \frac{1}{2} d_k^T C d_k + (C u_k + c)^T d_k + u_k^T c, \tag{A.6}$$

where $\frac{1}{2} u_k^T C u_k + u_k^T c$ is dropped because do not depend on d_k. A new optimization subproblem is defined:

$$\min_{d_k} \quad \frac{1}{2} d_k^T C d_k + (C u_k + c)^T d_k \in \mathbb{R}, \tag{A.7a}$$

$$\text{subject to:} \quad {B_i}^T u = 0, \quad i \in \mathcal{W}_k, \tag{A.7b}$$

where $\mathcal{W}_k$ is the working set, i.e., all equalities constraints and the inequalities constraints form $\mathcal{A}(u)$ imposed as equalities.

The null vector is a feasible solution of (A.7), so its objective value in (A.7a) must be larger than that of d_k, this way:

$$\frac{1}{2}d_k^T C d_k + (Cu_k + c)^T d_k < 0. \tag{A.8}$$

Since $d_k^T C d_k \geq 0$ by convexity (C is positive definite), this inequality implies $(Cu_k + c)^T d_k < 0$. Then,

$$f(u_k + \gamma d_k) = f(u_k) + \gamma(Cu_k + c)^T d_k + \frac{1}{2}\gamma^2 d_k^T C d_k < f(u_k), \tag{A.9}$$

for a $\gamma > 0$ sufficiently . From (A.8) the function $f(\cdot)$ is strictly decreasing along the direction d_k, whenever $d_k \neq 0$.

A.3 Proof of Theorem 2

This proof is based in [56]. Considering the SQP optimization problem given by:

$$\begin{aligned}
\min_{u} \quad & f(u) \ \in \mathbb{R}, && \text{(A.10a)}\\
\text{subject to:} \quad & c_i(u) = 0, \quad i \in \mathcal{E}; && \text{(A.10b)}\\
& c_i(u) \geq 0, \quad i \in \mathcal{I}. && \text{(A.10c)}
\end{aligned}$$

Given an iterative solution u_k in (A.10) generated by Algorithm 7 and $\tilde{H}_k(u_k)$ is the approximated Hessian using the BFGS iterative algorithm. Also, functions $f(\cdot)$ and $c(\cdot)$ are twice differentiable in a neighborhood of u with Lipschitz continuous second derivatives.

Given a strict local solution u in (A.10) where Definitions A.1.2 and A.1.4 hold. If $\| u_k - u \|$ and $\| \tilde{H}_k(u_k) - H(\mathcal{L}(u, \lambda)) \|$ are sufficiently small the following limit is satisfied

$$\lim_{k \to \infty} \frac{\| \mathsf{P}_k(\tilde{H}_k(u_k) - H(\mathcal{L}(u, \lambda))(u_{k+1} - u_k) \|}{\| u_{k+1} - u_k \|} = 0, \tag{A.11}$$

where $\mathsf{P}_k = I_n - \mathsf{A}_k^T(\mathsf{A}_k \mathsf{A}_k^T)^{-1}\mathsf{A}_k \in \mathbb{R}^{n \times n}$ and $\mathsf{A}_k^T = \begin{bmatrix} \nabla c_1(u_k) & \dots & \nabla c_i(u_k) \end{bmatrix}$, $i \in \mathcal{A}(u_k)$. Then, the iterates u_k converge superlinearly to u.