CIMAT

Center for Research in Mathematics

Master Thesis

Concurrent Design Optimization of Kinematically Complex Mechanisms

Author:
Salvador Botello-Aceves

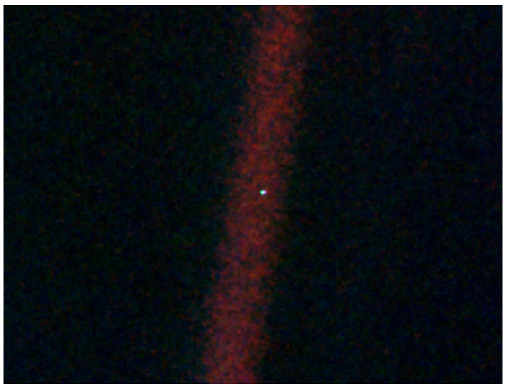
Supervisor:
Dr. S. Ivvan Valdez
Dr. Hectór M. Becerra

A thesis submitted in fulfilment of the requirements for the degree of Master in Science

in the

Computer Science Department

November 2016



Look again at that dot. That's here. That's home. That's us. On it everyone you love, everyone you know, everyone you ever heard of, every human being who ever was, lived out their lives. The aggregate of our joy and suffering, thousands of confident religions, ideologies, and economic doctrines, every hunter and forager, every hero and coward, every creator and destroyer of civilization, every king and peasant, every young couple in love, every mother and father, hopeful child, inventor and explorer, every teacher of morals, every corrupt politician, every "supersta", every "supreme leader", every saint and sinner in the history of our species lived there-on a mote of dust suspended in a sunbeam. The Earth is a very small stage in a vast cosmic arena. Think of the endless cruelties visited by the inhabitants of one corner of this pixel on the scarcely distinguishable inhabitants of some other corner, how frequent their misunderstandings, how eager they are to kill one another, how fervent their hatreds. Think of the rivers of blood spilled by all those generals and emperors so that, in glory and triumph, they could become the momentary masters of a fraction of a dot. Our posturings, our imagined self-importance, the delusion that we have some privileged position in the Universe, are challenged by this point of pale light. Our planet is a lonely speck in the great enveloping cosmic dark. In our obscurity, in all this vastness, there is no hint that help will come from elsewhere to save us from ourselves. The Earth is the only world known so far to harbor life. There is nowhere else, at least in the near future, to which our species could migrate. Visit, yes. Settle, not yet. Like it or not, for the moment the Earth is where we make our stand. It has been said that astronomy is a humbling and character-building experience. There is perhaps no better demonstration of the folly of human conceits than this distant image of our tiny world. To me, it underscores our responsibility to deal more kindly with one another, and to preserve and cherish the pale blue dot, the only home we've ever known.

Carl Sagan, Pale Blue Dot: A Vision of the Human Future in Space

CENTER OF RESEARCH IN MATHEMATICS

Abstract

Computer Science Department

Master in Science

Concurrent Design Optimization of Kinematically Complex Mechanisms

by Salvador Botello-Aceves

The problem of concurrent design optimization of a mechanism can be defined as finding optimal structural and control parameters, for a given objective function that describes the performance of a mechanism, at the same optimization process.

The main contribution of this work is to define, develop and test a general methodology that can generate optimal designs based on workspace and task requirements, such that they guarantee an adequate performance under a set of operating and joint constraints. This methodology intends to optimize any structure and control design, using any specified kinematic or dynamic model. Thus, general optimization methods non-dependent on mathematical characteristics of the objective function are used.

For this purpose, we test three families of evolutionary algorithms: a genetic algorithm, an evolution strategy and an estimation of distribution algorithm, for a set of objective functions. Therefore, we propose a categorization of the objective functions according to the differential equations order to be solved.

The second contribution of this work is the proposal of a parallel estimation distribution algorithm, which tries to minimize process communication without diminishing the algorithm performance. This proposal of parallelization is tested on the optimal mechanism objective function.

The optimal design results from each optimization method for every objective function are presented and discussed, allowing feedback and guidelines to improve the results or to address more objective functions as future work.

Acknowledgements

I acknowledge the support provided by the Consejo Nacional de Ciencia y Tecnologia (CONACyT), through its scholarship program, which has boosted my professional career.

To the Center for Research in Mathematics (CIMAT), for allowing me to acquire the necessary knowledge for my formation as a Master of Science within its facility and with its highly qualified research staff.

A particular acknowledgment to Dr. S. Ivvan Valdez Peña and Dr. Hector M. Becerra, for directing and guiding this thesis with patience and skill. Thank you both for teaching, counseling and challenging me in academic, professional and personal subjects.

To the committee members, Dra. Claudia Esteves Jaramillo and Dr. Eusebio E. Hernández Martinez, with their valuable suggestions, they contributed to the successful culmination of this work.

To all those, who in one way or another, contributed to the completion of the work presented in this thesis.

Contents

Abstract ii

Acknowledgements iii

List of Figures vii

List of Tables ix

Abbreviations x

Dedicatory xi

1 Introduction 1
- 1.1 Research motivation . . . 1
- 1.2 Addressed problems . . . 3
- 1.3 Thesis Objectives . . . 4
- 1.4 Thesis Structure . . . 5

2 Related work 6

3 Kinematically Complex Mechanisms 10
- 3.1 Definitions . . . 10
- 3.2 Generalized Mechanisms Kinematics . . . 11
- 3.3 Generalized Mechanisms Dynamics . . . 12
- 3.4 Feedback Control . . . 13

4 Optimization Methods 15
- 4.1 Evolutionary Algorithms . . . 15
  - 4.1.1 Omni-optimizer . . . 16
  - 4.1.2 Boltzmann Univariate Marginal Distribution Algorithm . . . 17
  - 4.1.3 Covariance Matrix Adaptation Evolutionary Strategy . . . 18
- 4.2 Parallelized Evolutionary Algorithms . . . 19
  - 4.2.1 Standard Master-Slave topology . . . 20
  - 4.2.2 Island topology network . . . 21

5 Objective functions 23

5.1 Objective 1: Minimum lengths subject to fulfill a workspace 24
5.2 Objective 2: Maximum regular workspace 25
5.3 Objective 3: Concurrent minimization of joint error with a kinematic control 26
5.4 Objective 4: Simultaneous optimization of joint error and consumed energy with a dynamic control 27
5.4.1 Objective 4.1: Minimal joint error 28
5.4.2 Objective 4.2: Minimal energy consumption 29

6 Proposed Methodology **30**
6.1 Proposed optimization methodology 30
6.2 Proposal of parallelization 31
6.2.1 Weighting in the formulae for parameter estimation 33
6.3 Concurrent Optimization Design for Mechanisms (*CODeMe*) Software . . 37

7 Results on case studies **39**
7.1 Case study: Anthropomorphic serial manipulator 39
7.1.1 Minimum lengths mechanism design subject to fulfill a workspace 40
7.1.2 Optimal concurrent mechanism design with a kinematic control . . 41
7.2 Case study: Delta parallel manipulator 44
7.2.1 Maximum regular workspace mechanism design 44
7.2.2 Optimal concurrent mechanism design with a kinematic control . . 47
7.3 Case study: Two-link serial manipulator 48
7.3.1 Minimal joint error mechanism design with a dynamic control . . . 49
7.3.2 Minimal energy mechanism design with a dynamic control 52
7.3.3 Optimal simultaneous mechanism design with a dynamic control . 54
7.3.4 Optimal concurrent mechanism design with a dynamic control . . 55
7.4 BFGS vs EAs 60
7.5 Proposed parallelization results 60

8 Conclusions and future work **62**
8.1 Conclusions 62
8.1.1 Conclusions: Optimization methods 62
8.1.2 Conclusions: Case studies 63
8.1.3 Conclusions: Optimization problems 64
8.1.4 Conclusions: Optimization results 64
8.2 Future Work 64

A Serial manipulators **66**
A.1 Serial manipulator kinematic 66
A.1.1 Forward Kinematics 67
A.1.2 Kinematics 68
A.1.3 Anthropomorphic manipulator kinematic 70
A.1.4 Two-link planar manipulator kinematic 71
A.2 Serial manipulator dynamic 72
A.2.1 Two-link planar manipulator dynamic 75

A.2.2 Counterweigth two-link planar manipulator dynamic 76

B Parallel manipulator **78**
B.1 Delta parallel manipulator 78
B.1.1 Inverse Kinematics 79
B.1.2 Jacobian matrix 81

C Controllers **83**
C.1 Proportional-integral-derivative controller 83
C.2 Adaptive proportional-derivative controller 84

Bibliography **86**

List of Figures

1.1 Statistics of actuated mechanisms worldwide. (a) Statistics of industrial actuated mechanisms used worldwide per year. (b) Statistics of 2013 and 2014, and projections to 2015-2018 of actuated mechanisms worldwide per year used in entertainment and household [1]. 1

1.2 Statistics of actuated mechanism worldwide per year. (a) Service mechanism sold in 2013 and 2014 used in differents fields. (b) Sold units in 2013 and 2014 used in main applications for professional uses [1]. 2

3.1 Graphical representation of a kinematically complex mechanism. 11

3.2 Basic structure of a feedback control system. 13

4.1 Schematic of a standard Master-Slave topology. 20

4.2 Schematic of an island topology network. 22

6.1 Graphic representation of the concurrent optimization method for kinematically complex mechanisms. 31

6.2 Visualization of behaviour of the average weighting using multivariate Gaussian distributions. (a) Wide randomized mean vector. (b) Near randomized mean vector. (c) Wide randomized covariance matrix. (d) Near randomized covariance matrix. (e) Average weighting. 34

6.3 Visualization of behaviour of the variance weighting using multivariate Gaussian distributions. (a) Same covariance matrix. (b) Linear attractor. (c) Light corner-attractor. (d) Heavy corner-attractor. (e) Light corner-attractors. (f) Heavy corner-attractors. 37

7.1 Case study: Anthropomorphic serial manipulator. (a) Schematic of the architecture of an anthropomorphic robot [2]. (b) Serial chain of an anthropomorphic robot. 40

7.2 Trajectory tracking control for the anthropomorphic robot. (a) Logarithmic absolute EEPE over time for P control. (b) P path tracking. (c) Logarithmic absolute EEPE over time for PD control. (d) PD path tracking. (e) Absolute EEPE integral for P and PD control. 42

7.3 Second case of study: Delta parallel manipulator. (a) Schematic of the architecture of the Delta robot [3]. (b) The i^{th} subchain of the Delta robot. 45

7.4 Workspace visualization. (a) 3D total and regular workspace. (b) Cross-section at 0.4137. (c) Cross-section at 0.5940. (d) Cross-section at 0.7742. 46

7.5 Trajectory tracking control for the Delta robot. (a) Logarithmic absolute EEPE over time for P control. (b) path tracking for P. (c) Logarithmic absolute EEPE over time for PD control. (d) path tracking for PD. (e) Absolute EEPE integral for P and PD control. 48

7.6 Case study: Two-link serial manipulator. Schematic of the architecture of the two-link planar manipulator. 49
7.7 Comparison of the best found and known control design for minimal error optimization. (a) Known optimal joints position. (b) Found optimal joints position. (c) Known optimal joints velocity. (d) Found optimal joints velocity. (e) Known optimal joints position error. (f) Found optimal joints position error. (g) Known optimal joints velocity error. (h) Found optimal joints velocity error. (i) Known optimal joints torque. (j) Found optimal joints torque. 51
7.8 Comparison of the best found and known control design for minimal energy optimization. (a) Known optimal joints position. (b) Found optimal joints position. (c) Known optimal joints velocity. (d) Found optimal joints velocity. (e) Known optimal joints position error. (f) Found optimal joints position error. (g) Known optimal joints velocity error. (h) Found optimal joints velocity error. (i) Known optimal joints torque. (j) Found optimal joints torque. 53
7.9 Comparison of the best found and known control design for simultaneous optimization. (a) Known optimal joints position. (b) Found optimal joints position. (c) Known optimal joints velocity. (d) Found optimal joints velocity. (e) Known optimal joints position error. (f) Found optimal joints position error. (g) Known optimal joints velocity error. (h) Found optimal joints velocity error. (i) Known optimal joints torque. (j) Found optimal joints torque. 56
7.10 Comparison of the best found and known design mechanism for concurrent optimization. (a) Known optimal joints position. (b) Found optimal joints position. (c) Known optimal joints velocity. (d) Found optimal joints velocity. (e) Known optimal joints position error. (f) Found optimal joints position error. (g) Known optimal joints velocity error. (h) Found optimal joints velocity error. (i) Known optimal joints torque. (j) Found optimal joints torque. 58
7.11 (a) Runtimes for the maximum regular workspace problem. (b) Runtimes for the minimal trajectory tracking error for a kinematic problem. (c) Speed up. (d) Efficiency. 61

A.1 Schematic of the architecture of an anthropomorphic robot. 71
A.2 Schematic of the architecture of a two-link planar manipulator. 72

B.1 3D structure of a parallel Delta manipulator. 79
B.2 Axis angles on the $i-th\ arm$ over point A_i. 79

C.1 A block diagram of a PID controller in a feedback loop. 84
C.2 A block diagram of an adaptive PID controller in a feedback loop. 85

List of Tables

7.1 Best solution for Eq. (5.1), links lengths $\{d_1, a_2, a_3, d_6\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$. 40
7.2 Best solution for Eq. (5.7), links length $\{d_1, a_2, a_3, d_6\}$, control gains $\{k_p, k_d\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$. 41
7.3 Results of the hypothesis test ($\mu_A < \mu_B$) with a 5% significancefor the optimization of an anthropomorphic serial manipulator, with the objective functions described in Eq. (5.1) and Eq. (5.7). 43
7.4 Best solution for Eq. (5.5), links lengths $\{a, b, d\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$. 46
7.5 Best solution for Eq. (5.7), links lengths $\{a, b, d\}$, control gains $\{k_p, k_d\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$. 47
7.6 Results of the hypothesis test ($\mu_A < \mu_B$) with a 5% significance for the optimization of a parallel Delta manipulator, with the objective functions described in Eq. (5.5) and Eq. (5.7). 49
7.7 Fixed conditions of the two-link planar manipulator. 50
7.8 Best solution for Eq. (5.10), control gains $\{k_p^{(j)}, k_d^{(j)}, k_i^{(j)}\}$ for j^{th} joint and objective value $\mathcal{F}_1$ and Eq. (5.11) ($\mathcal{F}_2$). 50
7.9 Best solution for Eq. (5.11), control gains $\{k_p^{(j)}, k_d^{(j)}, k_i^{(j)}\}$ for j^{th} joint and objective value $\mathcal{F}_2$ and Eq. (5.10) ($\mathcal{F}_1$). 52
7.10 Best solution for Eq. (5.9), using weights $\omega_1 = \omega_2 = 0.5$, control gains $\{k_p^{(j)}, k_d^{(j)}, k_i^{(j)}\}$ for j^{th} joint and objective value $\mathcal{F}^*$, Eq. (5.10)($\mathcal{F}_1$) and Eq. (5.11)($\mathcal{F}_2$). 55
7.11 Fixed conditions of the two-link planar manipulator. 57
7.12 Best solution for Eq. (5.9), using weights $\omega_1 = 0.8$ and $\omega_2 = 0.2$, control gains $\{k_p^{(j)}, k_d^{(j)}, a^{(j)}, b^j\}$ and counterweigth masses $m_\omega^{(j)}$ for j^{th} joint and objective value $\mathcal{F}^*$. 57
7.13 Hypothesis testing ($\mu_A < \mu_B$) with a 5% significance for the optimization of a two-link planar manipulator for different objective function described in Eq. (5.10), Eq.(5.11) and Eq.(5.9). 59
7.14 Best solution Eq. (5.10) ($\mathcal{F}_1$), Eq. (5.11) ($\mathcal{F}_2$) and for Eq. (5.9) ($\mathcal{F}_3$), using weights $\omega_1 = 0.5$ and $\omega_2 = 0.5$. 60

A.1 Example of an DH table. 68
A.2 DH table for a 6−DOF anthropomotphic manipulator. 71
A.3 DH table for a 2−DOF two-link planar manipulatror. 72

Abbreviations

BFGS	**B**royden-**F**letcher-**G**oldfarb-**S**hanno
BUMDA	**B**oltzmann **U**nivariate **M**arginal **D**istribution **A**lgorithm
CMA-ES	**C**ovariance **M**atrix **A**daptation **E**volutionary **S**trategy
CODeMe	**C**oncurrent **O**ptimization **De**sign for **Me**chanism
CRS	**C**ontrolled **R**andom **S**earch
DE	**D**ifferential **E**volution
DOF	**D**egrees **O**f **F**reedom
EA	**E**volutionary **A**lgorithm
EDA	**E**stimation **D**istribution **A**lgorithm
EEPE	**E**nd-Effector **P**osition **E**rror
ES	**E**volution **S**trategies
GA	**G**enetic **A**lgorithm
GAES	**G**enetic **A**algorithm-**E**volution **S**trategies
MPI	**M**essage **P**assing **I**nterface
NSGA	**N**on-dominated **S**orting **G**enetic **A**lgorithm
PEA	**P**arallelized **G**enetic **A**lgorithm
PID	**P**roportional-**I**ntegral-**D**erivative
PKM	**P**arallel **K**inematic **M**echanisms
PSO	**P**article **S**warm **O**ptimization
PUMA	**P**rogrammable **U**niversal **M**anipulation **A**rm
SA	**S**imulated **A**nnealing
SQP	**S**equential **Q**uadratic **P**rogramming

Dedicatory

To my parents **Salvador Botello Rionda** and **María Enriqueta Elizabeth Aceves Cervantes**:

> I am capable of achieving my goals in life because of your effort and sacrifice. Thanks to your example and inherited values, I am able to go forward, no matter how big the obstacle is. Keep in mind that with love, respect and admiration, I thank the two people who always watched over me and want the best for their children. Thank you for your unconditional support! I hope that my academic, professional and personal achievements will be felt as yours.

To my brothers **Jordi Botello Aceves** and **Elizabeth Botello Aceves**:

> Because, no matter how bad the situation was, we have always kept together. Thank you for being there when I have needed it and the long conversations we have shared.

To the memory of my grandfathers **Manuel Aceves Chavolla**† and **Salvador Botello Arias**†:

> Their absence has not diminished their remembrances and affection.

To my grandmothers and family:

> For their wise advice and stay by my side along the way.

To my girlfriend **Giselle Villafuerte Muñoz**:

> Because your love, affection and understanding have always driven me in life and always gets the best from me. Thanks for placing your confidence in me and being a brilliant confidant, friend and girlfriend. I love you!

To my friends, classmates and teachers:

> For their support, company and shared knowledge. We have shared great moments!

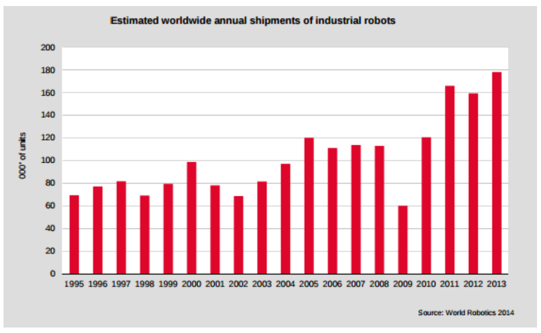
Chapter 1

Introduction

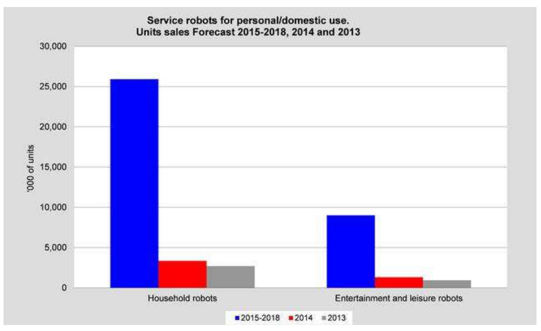
In this chapter, a motivation of the research project for concurrent design optimization of kinematically complex mechanisms is introduced. The addressed problems are briefly presented according to a proposed classification. Besides, the objectives of the work are stated and a brief description of the document structure is given.

1.1 Research motivation

Thanks to technological advances, the industrial sector is benefited every day with the use of specialized mechanisms for certain routine tasks with more complex control algorithms. Figure 1.1, shows the historical development of the use of actuated mechanisms in the industry since 1994 to 2008. Note that in the period 2001 to 2005, the use of robots in industries increased by 20% year over year. Actuated mechanisms installed in several companies in 2008 reached approximately $113,300$ units [1].



(a)

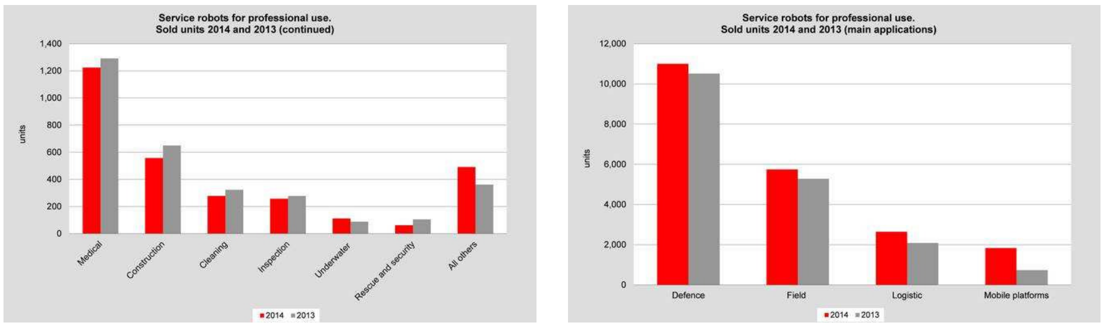


(b)

FIGURE 1.1: Statistics of actuated mechanisms worldwide. (a) Statistics of industrial actuated mechanisms used worldwide per year. (b) Statistics of 2013 and 2014, and projections to 2015-2018 of actuated mechanisms worldwide per year used in entertainment and household [1].

Figure 1.1 (a) shows the impact of the actuated mechanisms in our society, and how they have become a valuable economic asset in the industry, where year after year, they establish themselves as an indispensable tool. As well, figure 1.1 (b) shows some applications of the actuated mechanisms, and the corresponding projections of sales, which shows a substantial increase in sales in four years, implying that there is a wide range of opportunities for the development and optimization of such devices.

Although robots were early introduced for industrial applications, in the last years, the research and development have been moving to other applications, where scientific and social breakthroughs are expected. Thus, human-machine interaction is expected to increase in the future. Figure 1.2 shows the different socio-economic branches where actuated mechanism can be found, where we can notice that the amount of mechanism is very varied and the range of applications is wide.



(a) (b)

FIGURE 1.2: Statistics of actuated mechanism worldwide per year. (a) Service mechanism sold in 2013 and 2014 used in differents fields. (b) Sold units in 2013 and 2014 used in main applications for professional uses [1].

These benefits of robotics applications encourage the use of self-controlled mechanisms in action, inspire new purposes of use and demand improved designs. The design of a robot is subject to specific goals without breaking the mechanical and design constraints. As can be seen, the actuated mechanisms have a huge impact in our daily life and the amount of mechanisms in action is proportional to the amount of tasks that they are able to solve.

The selection of a mechanism to fulfill a certain objective and even choosing its geometrical structure and control parameters to perform a given task as best as possible is a difficult assignment. Thus, in this work we propose a generalized methodology that allows the optimization of a large set of actuated mechanism, based on an objective function that qualifies its performance. With this in mind, a flexible software based on the proposed methodology is developed. Libraries that support different actuated mechanism, objective functions and optimization methods are implemented into the application.

1.2 Addressed problems

Optimal concurrent mechanism design can be defined as *finding optimal structural parameters and control gains for a given objective function during the same optimization process, which is dependent on the kinematic or dynamic model of the mechanism.* The design parameters can be the number of links, links lengths, position and weight of masses, etc.

We seek to maximize the performance by optimizing the mechanisms to accomplish a desired task. This performance can be described as the energy consumption, the material used to built the mechanism, the error between the position of the end-effector and a fixed path, etc.

In this work, we differentiate three types of mechanical optimizations, according to the mathematical model describing the objective function. The proposed categorization of optimization problems is presented in the following three classes:

- **Static optimization problem**. The model to solve this problem consists of static relationships (zero order differential equations). Therefore, the equations of position and orientation (as homogeneous transformations) of the mechanism joints are used. A classic static objective function is the maximization of the reachable volume in the workspace.
- **Kinematic optimization problem**. In this case, the model to solve is a first-order differential equation, which involves computation of the velocities of the mechanism. For this objective function, the accelerations of the mechanisms are neglected, thus, the inertial properties of the mechanism are also obviated. An optimal design for trajectory tracking using a kinematic control is a common kinematic objective function. The links lengths or control gains are optimized to minimize the position and velocity errors of the end-effector or the joint variables. When both links lengths and control gains are optimized, it is called a concurrent optimization.
- **Dynamic optimization problem**. This is an acceleration-based objective function, where the inertial properties of the mechanism are now used to solve the second-order differential equations of motion. Inertial forces are considered within the model. Just as in the previous objective function, a trajectory tracking, but with a dynamical control, is a common dynamic objective function, where the relative inertial position of each link, the mass of each link, etc. can be also optimized.

These problems are directly related to the task or purpose of a mechanism. Hence, we might measure the mechanisms performance essentially by one of the following criterions: the largest regular region within a total workspace which can be covered by the manipulator, the error of the end-effector position and orientation with respect to desired values, the error between each joint position and desired values and the minimum energy consumption of every joint. In addition, the mechanism movements are subject to joint limits, collision and dexterity constraints.

1.3 Thesis Objectives

The main objective of this work is to propose a generic methodology to solve the concurrent optimization design of different kinematically complex mechanisms. These mechanisms are such that, due to a large number of degrees of freedom, redundancy or multiple kinematic chains, the kinematics is, typically, solved numerically rather than in a closed form.

In consequence of the mechanism complexity, the mathematical properties of an objective function such as gradient, smoothness and continuity, are not available or require a lot of computational or analytical effort. Thus, we propose to use gradient-free methods such like genetic algorithms and other evolutionary optimizers, which provide a flexible way of treating with the concurrent design optimization problem.

To carry out this objective it is necessary to lay the basic knowledge of optimization and modeling of robotic systems. The particular objectives of this thesis are the following:

1. A deep review of four different objective function models for the optimal design of mechanisms.
2. Evaluate a set of optimizers from three families: The Omni-optimizer [4], a well performed genetic algorithm (GA), the BUMDA [5] from the estimation of distribution algorithm (EDA) family, and the CMA-ES, an evolution strategy (ES).
3. Propose a parallelization method of an EDA to solve the objective functions.
4. Implement a software application based on the methodology to find the optimal design of the actuated mechanism.
5. Test the methodology on a simulated anthropomorphic serial manipulator, a Delta parallel manipulator [6] and a two-link planar manipulator using the software.
6. Compare the performance of optimization methods included in the software, in order to elucidate which one is best suited to solve such problems.

1.4 Thesis Structure

The organization of this thesis is as follows: related work in the field of actuated mechanisms optimization is summarized in Chapter 2. A general background of the kinematic and dynamic modeling of kinematically complex mechanisms extracted from literature is presented in Chapter 3. In Chapter 4, we briefly review the three state-of-the-art evolutionary algorithms used for approximating the solution to the optimum objective functions. In Chapter 5, we introduce and discuss the different models for optimum mechanism design. In Chapter 6, the proposed methodology for the concurrent optimization design is introduced, as well as the proposed parallelization of the estimation distribution algorithm. In Chapter 7, we present three cases of study for actuated mechanisms along with the results of applying the different optimization methods to the objective functions. Finally, in Chapter 8, we present the final discussion and concluding remarks.

In order to make this document self-contained, we have included three appendices about robot modeling and control.

Chapter 2

Related work

In this section, we present the most representative works for the optimization of actuated mechanism using different optimization methods, optimization problems and mechanisms.

Let us recall the proposed classification of the optimization problems introduced in Section 1.2, where the main difference between optimization problems is the differential equation order that has to be solved. Firstly, let us present the static optimization problem that can be found in literature. These problems can be described as optimization problems where there is no differential equation to solve. Therefore, it only depends on the kinematic equations expressed as homogeneous transformations, commonly, maximizing a desirable workspace and considering the joint constraints, collision avoidance and dexterity constrains.

In [7–9], a method for genereting optimal manipulator geometric structures based on task requirements is introduced and tested. To ensure a successful accomplishment of the task, the objective function is subject to user defined joint constraints. The methodology consists in reaching a path, by computing its inverse kinematics, which determines if the manipulator can reach and orient the end-effector at the task point within the set of joint limits. Using Simulated Annealing (SA), the method optimizes the geometrical parameters of a manipulator with six degrees of freedom (DOF).

Similar to the previous work, in [10], a method to find the optimal geometric structure of a non-redundant 6-DOF serial manipulator, from task descriptions is presented. This work uses the same objective function as in [9], but adding as a constraint the determinant of the Jacobian matrix, which is deeply related to the dexterity of the mechanism. SA is applied to optimize the geometric structure of the 6-DOF serial manipulator.

The work in [11] deals with the optimal kinematic synthesis of parallel manipulators. The optimization problem is formulated as finding the manipulator geometry that maximizes an effective regular workspace. The concept of *effective regular workspace* is defined in function of a dexterity index, computed as the inverse condition number of the Jacobian matrix, it is used to characterize the effectiveness of the workspace. This improves both requirements of the workspace: shape and quality. In this case, a controlled random search (CRS) technique is applied to optimize the kinematic synthesis of four parallel manipulators: different configurations of the five-bar parallel linkage, a 3-RRR planar parallel manipulator, a Delta robot and a Stewart-Gough platform. In [12], the same procedure is applied to optimize a Stewart-Gough platform using a classic genetic algorithm (GA).

In [13], the same procedure than in [11] is used to find the optimal design parameters of a Delta robot and a Stewart-Gough platform. They use the same direct search method than in [11], adding a collision avoidance constraint, which is simplified by adjusting the limits of the joints. Finally, in [14], the authors compare several optimization methods to find the optimal kinematic synthesis of two parallel manipulators. A sequential quadratic programming (SQP), a CRS, a GA, a differential evolution (DE) and a particle swarm optimization (PSO) are applied to optimize the geometric parameters of a Delta robot and a Stewart-Gough platform. Based on their conclusion, the SQP with multiple initial points performs the best for the Delta robot case, while for the Stewart-Gough platform, SQP performs the worst and DE and PSO are the best choices, but the algorithms are compared without a statistical test. Since the implemented optimization methods are stochastic algorithms, this comparison does not allow us to elucidate which algorithm delivers the best approximation to the optimum and how frequently.

A design method that aims to minimize the cost of a serial robot manipulator that satisfies a target workspace and payload constraints is presented in [15]. In this case, the problem is to find the cheapest serial robot that can carry a specific amount of load in a space at least as large as a defined rectangular prism, where the cost is related to the lengths of the manipulator. The study cases are a *2-R planar robot* and a *6-R arm with spherical wrist*, where *n-R* represents n revolute joints, using a variant of a SA with random restarts. A similar approach is implemented in [16], where the previous method is applied using a randomized linear search method, to optimize the Delta robot.

Regarding the second class of optimization problems, kinematic optimization problems, the basic idea is to solve a first-order differential equation that represents relationships between velocities. In this case, a typical problem is that the mechanism tracks a trajectory imposed by the user, which seeks to minimize the error between the trajectory and the position and orientation of the end-effector of the mechanism or the positions of the joints and their desired value. This type of problem typically involves solving the kinematics of a redundant mechanism, applying collision avoidance, position constraints and speed constraints. It is very common to use a variation of a proportional-integral-derivative control [17, 18]. Therefore, the geometric parameters, as well as the control parameters of the mechanisms, are the optimizable variables. Some authors do not consider geometric parameters for the optimization [18, 19].

To perform trajectory tracking, it is necessary to use a control technique which feeds back to the actuators according to the tracking error. A set of differential equations describes the system, and the control law is a function of the error and a set of parameters named control gains. A classical control technique in robotics is the PID (Proportional-Integral-Derivative) controller and its variants as P (only proportional) or PD (Proportional-Derivative) controllers [20]. This control technique uses a factor (control gain) for each of the control actions (proportional, integral and derivative). Hence, the control gains define how well the trajectory is tracked and how much energy is needed for this purpose. The tuning of the control gains in order to achieve an optimum performance is a difficult task, commonly, testing and empirical knowledge from the user are applied to set the control gains.

In [21], the optimum design of the control parameters of a serial manipulator is approached. Only kinematics is considered in a task-based problem. The task to perform is defined by a set of target points in the workspace which must be reached by the end-effector. In order to approximate the solution to this problem, a memetic algorithm which uses a tunneling algorithm is used. A similar scheme has been proposed in [18], where simultaneous design optimization of a planar manipulator and a non-linear gain proportional-derivative (PD) controller for a trajectory tracking task is presented. They used a hybrid evolutionary algorithm (GAES), which is formulated by combining the concepts from two well-known members of evolutionary algorithms family, GA and evolution strategies (ES).

The same last procedure is used in [17], to optimize a redundant 7-DOF spatial manipulator. A task-priority redundancy resolution technique introduced and the SQP method is applied to optimize the control parameters of the mechanism. In [19], a dual neural network is used to solve the inverse kinematics problem of a 5-DOF planar manipulator and a 7-DOF spatial manipulator. In this method, the inverse of the Jacobian matrix is approximated using a positive definite weighting matrix, computed using a recursive neural network, which is trained in order to minimize a parametric program. Once the inverse of the Jacobian matrix is computed, the joint positions are updated using the Euler step method.

For dynamic optimization problems, where a second order differential equation must be solved. It can be separated and solve with at least two first order differential equations, using a variable change, thus, a numerical method with higher precision is required to reduce the numerical error, due to its accumulation by solving several first order differential equations. Commonly, the mechanism has to track a time-dependent trajectory, which can be given by a time-dependent function or a set of points which define a function via interpolation methods. Typically, the aim is to minimize the error between the desired and the executed trajectory or to minimize the energy consumed by every joint through the whole trajectory, subject to collisions and dexterity constrains.

A method for the energy consumption optimization of robotic systems is presented in [22]. Two case studies are simulated in order to assess the validity of the proposed procedure: a PUMA 560 (spatial serial manipulator) and a PKM 3D (spatial parallel manipulator). Based on an electro-mechanical model of industrial robots, where the applied torque is used to compute the energy consumption in function of energetic losses, due to friction and viscous forces, and electrical efficiencies, a trajectory time scaling control is applied to track the desired path.

A simplified dynamical model is used in [23], where a numerical optimization technique for simultaneous optimization of design parameters of a two-link planar rigid manipulator with a non-linear gain PD controller is presented. A $(\mu + \lambda)$-ES is applied to minimize simultaneously the torque applied by each joint and the error between the joints and the desired position.

In a similar vein, the optimization of a PID controller of a two-link planar manipulator through Non-dominated Sorting Genetic Algorithm II (NSGA-II) is presented in [24]. The optimization method is a computationally efficient GA with a selection method based on classes of dominance of all the solutions. In this case, the position error and the applied torque of every joint are minimized. A similar approach is presented in [25], where a concurrent design of a mechatronic system is presented. A multi-objective optimization using a GA with continuous and discrete variables is conducted to minimize the torque applied by the motors and the fluctuation of the input velocity. The control gains of a non-lineal PD controller applied to a planar parallel manipulator are optimized.

Chapter 3

Kinematically Complex Mechanisms

In this chapter, we present a brief description of the mathematical modeling and control aspect of mechanisms from a general point of view. For more details, the reader is referred to specialized literature [26–28]. In particular, the differential kinematics and dynamics of the kinematically complex mechanism are presented, which gives the relationship between the joint velocities, the linear and angular velocities and the joint actuator torques of the end-effector, respectively.

3.1 Definitions

Let us define the general terms used in the optimization models described in Chapter 5. Consider a mechanism, as shown in Fig. 3.1, with m joints and a actuators and d DOF of the end-effector. The mechanism is defined by a set of design parameters $\boldsymbol{\alpha} \in \mathbb{R}^p$, for instance, the lengths of the links, the relative position of each link, the relative arrangement between each axis, the size and shape of the end-effector, etc.

Let us define a target workspace $\mathbf{X} \in \mathbb{R}^{d \times n}$ as a set of Cartesian coordinates of position and orientation in a reference framework, where n is the number of points. Notice that these points can represent a whole target workspace or a path which is a subset of the whole workspace.

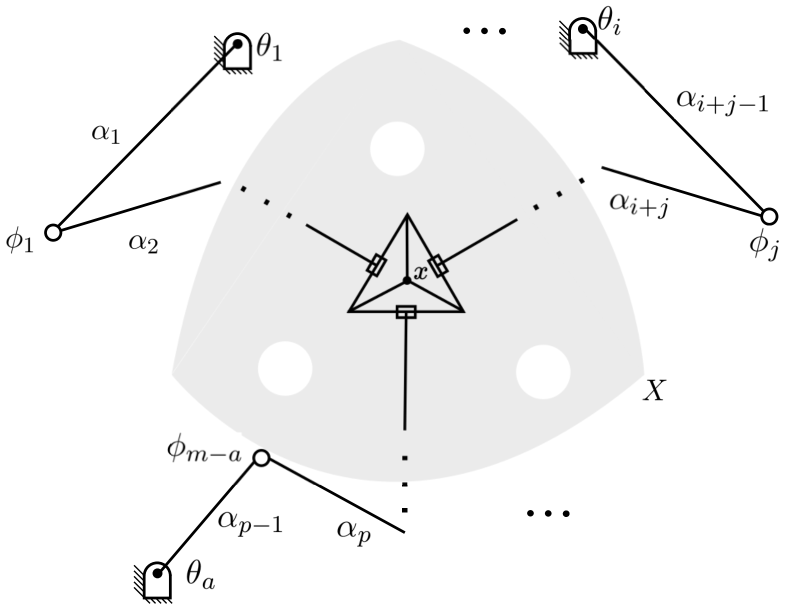


FIGURE 3.1: Graphical representation of a kinematically complex mechanism.

3.2 Generalized Mechanisms Kinematics

The generalized kinematics of a mechanism can be expressed as a relationship between the Cartesian velocities of the end-effector with respect to a base reference frame and the joint velocities through a matrix operator named Jacobian matrix. It is possible to compute the Jacobian matrix via differentiation of the direct kinematics function with respect to the joint variables if the end-effector pose is expressed in a minimal representation in the operational space.

For a given point $\mathbf{X}_k$ and parameters $\boldsymbol{\alpha}$, using the inverse kinematics, we can denote a set of values for the actuated joint variables $\boldsymbol{\theta} \in \mathbb{R}^a$, and passive joint variables $\boldsymbol{\phi} \in \mathbb{R}^{m-a}$, as in Equations (3.1) and (3.2).

$$\theta_i = \theta_i(\mathbf{X}_k, \boldsymbol{\alpha}), \quad i = 1, \ldots, a, \tag{3.1}$$

$$\phi_j = \phi_j(\mathbf{X}_k, \boldsymbol{\alpha}), \quad j = 1, \ldots, m-a. \tag{3.2}$$

Let us define $\mathbf{x} \in \mathbb{R}^d$ as the position and orientation of the end-effector. Thus, the velocities of the end-effector are given by:

$$\dot{\mathbf{x}} = \mathbf{J}\dot{\boldsymbol{\theta}} \tag{3.3}$$

where $\mathbf{J} = \mathbf{J}(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\alpha}) \in \mathbb{R}^{d\times a}$ is the Jacobian matrix, which relates the angular velocities $\dot{\boldsymbol{\theta}}$ with the Cartesian velocities $\dot{\mathbf{x}}$ of the end-effector.

The Jacobian constitutes one of the most important tools for manipulator characterization. In fact, it is useful for finding *singularities*, analyzing *redundancy* and determining *inverse kinematics algorithms* [29, 30]. More details about the Jacobian computation are given in Appendixes A and B.

3.3 Generalized Mechanisms Dynamics

The derivation of the dynamic model of a mechanism plays an important role for simulation of motion, analysis of manipulator structures and design of control algorithms. It provides a description of the relationship between the joint actuator torques and the motion of the structure. Using a Lagrange formulation, the equations of motion can be derived in a systematic way, independent of the reference coordinate frame from each link. The Lagrangian $\mathcal{L}$ of the mechanical system can be defined as a function of the generalized coordinates $\boldsymbol{\theta}$, as $\mathcal{L}(\dot{\boldsymbol{\theta}}, \boldsymbol{\theta}) = \mathcal{T}(\dot{\boldsymbol{\theta}}, \boldsymbol{\theta}) - \mathcal{U}(\boldsymbol{\theta})$, where $\mathcal{T}$ and $\mathcal{U}$ denote the total kinetic and potential energy of the system, respectively.

First, let us assume that the mechanism is formed by a set of kinematic chains. Each kinematic chain has a reference frame associated with each link i . The Lagrange equations are expressed by

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\theta}}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}\right)^T = \boldsymbol{\xi} \tag{3.4}$$

where $\boldsymbol{\xi} \in \mathbb{R}^a$ are the generalized forces associated with the generalized coordinates $\boldsymbol{\theta}$. Given by the nonconservative forces, i.e., the joint actuator torques, the joint friction torques, as well as the joint torques induced by end-effector forces at the contact with the environment.

Taking the derivatives required by Lagrange equations, Eq. (3.4), and recalling that $\mathcal{U}$ does not depend on $\dot{\boldsymbol{\theta}}$ yields,

$$\mathbf{B}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\dot{\boldsymbol{\theta}}, \boldsymbol{\theta})\dot{\boldsymbol{\theta}} + \mathbf{F}_v\dot{\boldsymbol{\theta}} + \mathbf{F}_s\mathbf{sgn}(\dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) = \tau - \mathbf{J}^T(\boldsymbol{\theta})\mathbf{h}_e \tag{3.5}$$

where $\mathbf{B}(\boldsymbol{\theta}) \in \mathbb{R}^{a\times a}$ is a positive definite symmetric matrix called matrix of inertia. $\mathbf{C}(\dot{\boldsymbol{\theta}}, \boldsymbol{\theta}) \in \mathbb{R}^{a\times a}$ are the quadratic velocity terms, where the centrifugal and the Coriolis effects are represented. $\mathbf{G}(\boldsymbol{\theta}) \in \mathbb{R}^a$ are the configuration-dependent terms, which represents the moment generated by the presence of gravity. A deeper analysis can be found in [26].

The rest of the terms in Eq. (3.5) correspond to nonconservative forces applied to the mechanism joints. $\mathbf{F}_v \in \mathbb{R}^{a\times a}$ denotes the diagonal matrix of viscous friction coefficients.

The term $\mathbf{F}_s\mathbf{sgn}(\dot{\boldsymbol{\theta}})$ models the static friction torque by symplifying them to the Coulomb friction torques, where $\mathbf{F}_s \in \mathbb{R}^{a\times a}$ is a diagonal matrix and $\mathbf{sgn}(\dot{\boldsymbol{\theta}}) \in \mathbb{R}^{a}$ is the sign functions vector, where each components is given as the sign functions of the single joint velocity.

If the end-effector of the mechanism is in contact with the environment, a portion of the actuation torques is used to balance the torques induced at the joints by the contact forces. Such torques are given by $\mathbf{J}^T(\boldsymbol{\theta})\mathbf{h}_e$, where $\mathbf{h}_e$ denotes the vector of force and moment exerted by the end-effector on the environment.

Some joint dynamic couplings and nonconservative forces may be reduced or zeroed when designing the structure, so as to simplify the control problem. In this work, the nonconservative forces in the dynamic models presented in Appendixes A and B are neglected.

3.4 Feedback Control

In this work, different controllers for actuated mechanical systems are used, thus, a generalized and brief introduction to the control theory is presented.

The dictionary of mathematics [31] describes a controller as follows: *In a series of operations, information gained at one stage can be used to modify later performances of that operation. This is known as feedback and enables a control system to check and possibly adjust its actions when required.*

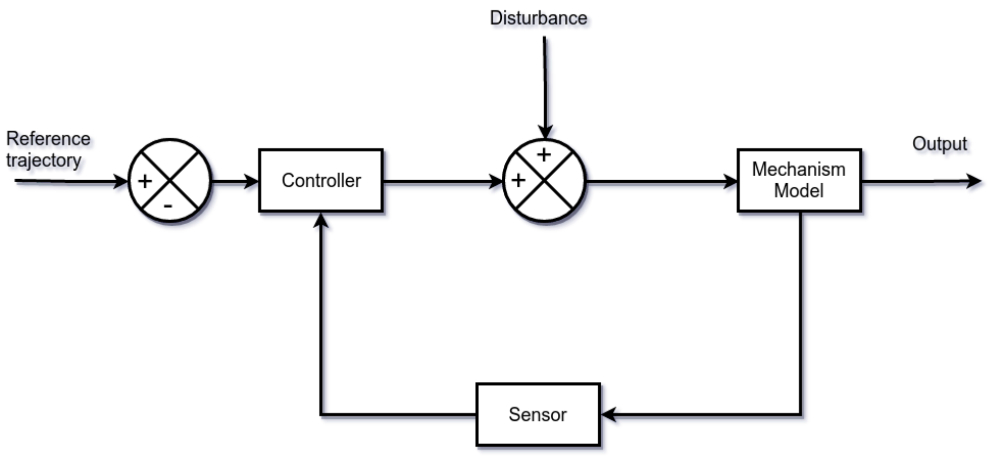


FIGURE 3.2: Basic structure of a feedback control system.

In a robotic mechanism, a control scheme is responsible for adjusting velocities, forces or torques applied to the actuated joints of the mechanism, to perform the desired task.

To change the behavior of the system, the control must be designed on the basis of a control law with feedback as shown in Fig. 3.2.

The general models of the mechanisms, shown in the previous section, may be represented as a finite number of coupled ordinary differential equations. In vector form they can be described as:

$$\dot{\mathbf{q}} = \mathbf{f}(t, \mathbf{q}, \mathbf{u}) \tag{3.6}$$

where $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ are the state variables, and $\mathbf{u} = (u_1, u_2, \ldots, u_a)$ are the input variables. Typically, the state variables represent the positions and velocities of the joints of the robot, while the input variables represent the velocities, forces or torques that can be applied to the robot through actuators. The control scheme is responsible for applying the inputs required for the mechanical structure to perform the desired behavior.

The control objective is to design a control law with feedback as:

$$\mathbf{u} = \gamma(t, \mathbf{q}), \tag{3.7}$$

such that the closed loop system, given by

$$\dot{\mathbf{q}} = \mathbf{f}\left(t, \mathbf{q}, \gamma(t, \mathbf{q})\right), \tag{3.8}$$

delivers solutions with certain characteristics, such as stability of a desired equilibrium point.

Chapter 4

Optimization Methods

In order to approximate the optimum solutions of the optimization problems, we must use an optimization method. Optimization methods for non-linear problems can be categorized as gradient-based and direct search methods. A gradient based method requires a continuous, derivable and convex objective function as well as analytical or numerical work in order to compute derivatives, while direct search methods only require of function evaluations. Evolutionary Algorithms (EAs) are direct search methods, which only require the evaluation of the objective function (and constraints) for a given candidate solution. These global optimization methods have been widely used for hard optimization of non-linear problems, such as synthesis of actuated mechanisms and optimal control [8, 10, 32].

4.1 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are based on the principles of biological evolution, a general framework can be stated as follows: first they generate a population of candidate solutions, then the best solutions are selected. By using the selected set, a variation operator is applied to generate a new set of candidate solutions, these solutions replace the current population. Usually, the best solution is preserved trough generations and the process is repeated until the stopping criterion is met.

The EAs used in this work can be classified into three families: Estimation of Distribution Algorithms (EDAs) [33], Evolution Strategies (ESs) [34], and Genetic Algorithms (GAs) [35]. Nevertheless, all of them can be classified as evolutionary algorithms but each of them has a particular way of working. The Omni-optimizer is the implementation of a multi-objective GA, re-combinates the most promising candidates selected using binary tournament. The exploration is maintained via a mutation operator. The Boltzmann Univariate Marginal Distribution Algorithm (BUMDA) estimates a probability distribution by using the best candidates, the better a candidate is, the higher the weight of such candidate in the estimation formula. Thus, the resulting probability functions favor to sample the best regions already known. The Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) uses a reproduction operator which favors promising directions. The Omni-optimizer and BUMDA share the characteristic of sampling the regions where the best candidates are, while the CMA-ES samples in *directions* where the best candidates were generated. A brief introduction to each of the algorithms is given in the rest of the section. See the corresponding references [5, 36, 37] for more information.

4.1.1 Omni-optimizer

The Omni-optimizer [36] is a general optimization GA, which is used, in this case, to solve a single objective problem. Nevertheless, it is an optimizer that can be applied to a wide range of problems from mono-objective to multiobjective with and without constraints in discrete and continuous domains. In our problem, we used it as a simple GA with simulated binary crossover (SBX) [38] and polynomial mutation. These are the same operators than those used in the NSGA-II [39], which probably, is the most widely used GA in the last ten years.

Algorithm 1 Omni-optimizer (NSGA-II) Pseudocode

```
1: $P_0 \leftarrow \mathcal{U}(\min^{(i)}, \max^{(i)})$
2: $\mathcal{F}_0 \leftarrow f(P_0)$
3: Sort $P_0$ based on the non-dominated solutions
4: $Q_0 \leftarrow$ tournament selection, recombination and mutation operators($P_0$)
5: $\mathcal{G}_0 \leftarrow f(Q_0)$
6: $t \leftarrow 0$
7: while stopCrit $\neq$ true do
8:     $R_t = P_t \cup Q_t$
9:     Sort $R_t$ based on the non-dominated solutions
10:    Generate all non-dominated front $\mathcal{F}_t$
11:    Select $P_{t+1}$ by crowding distance
12:    $\mathcal{F}_{t+1} \leftarrow f(P_{t+1})$
13:    Sort $P_{t+1}$ based on the non-dominated solutions
14:    $Q_{t+1} \leftarrow$ tournament selection, recombination and mutation operators($P_{t+1}$)
15:    $\mathcal{G}_{t+1} \leftarrow f(Q_{t+1})$
16: end while
17: return $P_t^{(1)}$
```

In Algorithm 1, we reproduce the general algorithm of the Omni-optimizer: in steps 1-2 the population is randomly initialized and evaluated from a uniform distribution.

Then, the population is sorted and selected via the Pareto ranking, and a population of new candidates is created and evaluated in steps 3-5. In steps 8-10 the previous and new populations are joined, to be sorted and selected using the Pareto ranking. The solutions with the same rank are selected by crowding distance and sorted once again to generate and evaluate a new population in steps 11-15. Steps 8-15 are repeated until a stopping criterion is met.

4.1.2 Boltzmann Univariate Marginal Distribution Algorithm

The Boltzmann Univariate Marginal Distribution Algorithm (BUMDA) [5] is an EDA that uses a univariate Gaussian model to approximate the Boltzmann distribution, whose energy function is related to the objective function. This means that the better a candidate is, the probability to sample in the same region increases. The mean and variance parameters of the Gaussian model are derived from the analytical minimization of the Kullback-Leibler divergence. Pseudocode of the BUMDA is shown in Algorithm 2.

Algorithm 2 Boltzmann Univariate Marginal Distribution Algorithm (BUMDA) Pseudocode

```
Require: D=dimensions of the problem. $n_{pop}$ =user given population size.
 1: $X \leftarrow \mathcal{U}(\min^{(i)}, \max^{(i)})$
 2: $\hat{\mathcal{F}} \leftarrow f(X)$
 3: Sort $X$ according to the objective function
 4: $n \leftarrow n_{pop}$, $\theta_t \leftarrow \hat{\mathcal{F}}_n$, $t \leftarrow 0$
 5: while stopCrit $\neq$ true do
 6:     $\mu_t^{(i)} \leftarrow \frac{\sum_{j=0}^{n} x_j^{(i)} \hat{\mathcal{F}}(x_j)}{\sum_{j=0}^{n} \hat{\mathcal{F}}(x_j)}$
 7:     $v_t^{(i)} \leftarrow \frac{\sum_{j=0}^{n} \hat{\mathcal{F}}(x_j)(x_j^{(i)} - \mu_i)^2}{1+\sum_{j=0}^{n} \hat{\mathcal{F}}(x_j)}$
 8:     $x_{best}$=Elitism$(X, \hat{F})$
 9:     $X \leftarrow (\mathcal{N}(\mu_t, v_t) \cup x_{best})$
10:     $\hat{\mathcal{F}} \leftarrow f(X)$
11:     Sort $X$ according to the objective function
12:     $n \leftarrow \min\left(\max(k|\theta_t < \mathcal{F}_k), \frac{1}{2} \cdot n_{pop}\right)$
13:     $\theta_t \leftarrow \hat{\mathcal{F}}_n$
14:     $t \leftarrow t + 1$
15: end while
16: return $x_1$
```

In steps 1-2, the initial population X is sampled from a uniform distribution and it is evaluated on the objective function f. Considering that the BUMDA is a maximization algorithm and we are using it to solve a minimization problem, the vector with the original objective function values F is transformed to $\hat{F} = 1.0 - (F - F_{max})$, where F_{max} is the minimum value in the vector $-F$. The population is sorted according to the objective function in step 3. Then, using the candidates with objective function value greater or equal to θ_t, mean and variance are computed for each dimension independently in steps 6-7. In step 8-10, the best candidate x_{best} is preserved through generations and the new population is sampled, from a normal distribution which uses the computed mean and variance. The threshold θ_t is augmented each iteration to ensure the improvement of the selected set. Steps 6-14 are repeated until a stopping criterion is met.

4.1.3 Covariance Matrix Adaptation Evolutionary Strategy

The Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [37] uses a multivariate Gaussian model. Parent solutions are used to determine the size, position and orientation of a Gaussian distribution used to sample the children candidate solutions. One of the most important characteristics of the CMA-ES is that the orientation of the multivariate Gaussian model directs the search to promising regions, in a kind of descent path. The Gaussian mean is computed as the addition of a weighted sum of the difference between the current population (parents) and the current mean.

Algorithm 3 Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) Pseudocode

```
1: Set $\lambda \leftarrow$number of samples per iteration, at least two, generally $> 4$
2: Initialize $m, \sigma, C = I, p_\sigma = 0, p_c = 0$
3: $t \leftarrow 0$
4: while stopCrit $\neq$ true do
5:     for $i = 1 : \lambda$ do
6:         $x_i \leftarrow \mathcal{N}(m, \sigma^2 C)$
7:         $\mathcal{F}_i \leftarrow f(x_i)$
8:     end for
9:     $x_{1\ldots\lambda} \leftarrow x_{s(1)\ldots s(\lambda)}$ with $s(i) = argsort(f(1,\ldots,\lambda), i)$
10:    $m' = m$
11:    $m \leftarrow update\_m(x_1, \ldots, x_\lambda)$
12:    $p_\sigma \leftarrow update\_ps(p_\sigma, \sigma^{-1}C^{-1/2}(m - m'), ||p_\sigma||)$
13:    $p_c \leftarrow update\_pc(p_c, \sigma^{-1}(m - m'), ||p_\sigma||)$
14:    $C \leftarrow update\_C(C, p_c, (x_1 - m')/\sigma, \ldots, (x_\lambda - m')/\sigma)$
15:    $\sigma \leftarrow update\_sigma(\sigma, \ldots, ||p\sigma||)$
16: end while
17: return $x_1$
```

In Algorithm 3, steps 1-3 initialize the multivariate Gaussian distribution and the algorithm parameters. In steps 5-7, the new candidate solutions are generated from the multivariate Gaussian distribution and evaluated on the objective function f. In step 9 the population is sorted for selecting the best solutions. The previous mean is stored, then the new mean, the isotropic path p_σ and the anisotropic path p_c are computed in steps 10-13. Lastly, in steps 14-15 the new covariance matrix is computed and the isotropic and anisotropic path are used to modify the new covariance matrix for guiding the search toward the maximum improvement directions as the maximum variance direction. Steps 5-15 are repeated until a stopping criterion is met.

4.2 Parallelized Evolutionary Algorithms

In the previous section, a general description of EAs is presented and as it can be seen, population-based EAs are highly parallelizable. Thus, EAs could benefit from parallel computing in order to reduce the computational time or to improve the quality of the solution.

As mentioned in [40], several EAs have been parallelized, with the main objective of obtaining algorithms with an execution time much lower than that of their sequential counterparts in order to tackle more complex problems [41–44].

A common way to parallelize an EA is by parallelizing the fitness function, which decreases the computational time in the evaluation. But, in some cases, the fitness function cannot be parallelized, e.g., the fitness function is time dependent. That is to say, requires a loop which performs time step calculations, consequently iterations $i + 1$ can not be independently computed of iteration i. Another highly efficient way to parallelize an EA is by parallelizing the evaluation of the fitness function of every candidate, thus a communication pattern between threads/processes has to be introduced in order to distribute the workload among process and to gather the results from each process/thread to a single one.

There are two main paradigms of parallel computation: shared and distributed memory. The first considers a set of processing units directly connected to a single memory, in such a way that any processing unit can access to any memory location, in consequence, the message passing between processing units is not necessary. Even if the processing units access the same memory location, they can not write or read at the same time, thus, consistency of the memory data must be carefully taken into account. The second considers a set of processing units with their own memory, hence they must communicate via messages in order to collaborate. On one hand, the second paradigm requires a higher coding effort and introduces an overhead due to the communication operations. On the other hand, the total memory and processing capacities of the system is the sum of the processing capacity and memory of all the nodes.

In addition, when all the processing units intend to access the same memory by a single data bus, the scalability of the system is constrained by the data bus capacity, while distributed memory systems scale as much as the interconnection of the nodes is possible.

In practice, distributed memory systems very often use thousands of processing units, while the shared memory systems, usually, have one or two dozens of processing units.

In most of the Parallelized Evolutionary Algorithms (PEAs) with distributed memory, two main communication protocols are used, the master-slave and the island network. Their advantages and disadvantages are discussed in sections below.

4.2.1 Standard Master-Slave topology

On the master-slave topology communication, shown in Figure 4.1, a device or process known as the master, controls one or more other devices or processes, known as slaves. Once the master-slave relationship is established, the direction of control is always from master to slaves. In Algorithm 4, a pseudocode for the EAs parallelization communication protocol is shown for master and slaves. Commonly, the fitness of every candidate solution in the new population are evaluated by the slaves, and the results are sent back to the master, where the remaining EA operations are executed.

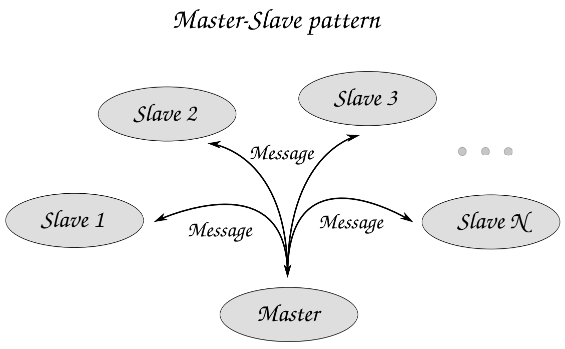


FIGURE 4.1: Schematic of a standard Master-Slave topology.

As an advantage of this paradigm, once the candidates in the current population are evaluated, the whole population is used to represent the location of the potential new candidates. The send/receive operations between the master and slave delay the parallel process. Let us assume a total time of communication for one generation of an EA using an ideal master-slave protocol, where the last slave takes the same time to evaluate the subpopulation as the master to receive the evaluations from the other slaves, given as:

$$t_{total} = t_f \phi_n + \sum_{i=1}^{S} (\beta \phi_i l_i + L) + (\beta \phi_s l_f + L) \tag{4.1}$$

where ϕ_n, ϕ_i and ϕ_s are the number of candidate solutions in the population, the number of evaluated candidate solutions in process i and the number of evaluated candidate solutions by the final process S, respectively. t_f is the evaluation time of the objective function, β is the system bandwidth, L is the system latency, l_i is the length of a candidate solution in bytes and l_f is the length of the evaluation in bytes. Eq. (4.1) is the time model of the data transfer for a master-slave process with an imbalance of the number of candidate solutions in S slaves processes. Since β and L are data given by the computational resources, let us assume that each process evaluates the same number of candidate solutions $\phi_p = \phi_n/S$, thus $\phi_i = \phi_p \ \forall \ i = 1, \dots, S$, and the bandwidth and latency are constant and equal values within each process:

$$t_{total} = t_f \phi_p + S\left(\beta \phi_p l_i + L\right) + \left(\beta \phi_p l_f + L\right) \tag{4.2}$$

where $l_i = l_i(d)$, $l_f = l_f(d)$. Notice that the total communication time is proportionally affected by the dimension of the problem d.

Algorithm 4 Standard Master-Slave topology for evolutionary algorithms pseudocode

```
MASTER
 1: Initialize the parameters of the probability distribution
 2: while stopCrit $\neq$ true do
 3:     Simulate a new population
 4:     for $i = 1 : \mathcal{S}$ do
 5:         Send to SLAVE$_i$ subpopulation
 6:     end for
 7:     for $i = 1 : \mathcal{S}$ do
 8:         Receive from SLAVE$_i$
 9:     end for
10:     Compute the probability distribution parameters
11: end while
```

```
SLAVE$_i$
1: while stopCrit $\neq$ true do
2:     Receive from MASTER
3:     Evaluate subpopulation
4:     Send evaluations to MASTER
5: end while
```

4.2.2 Island topology network

In the communication protocol for the island network topology, shown in Figure 4.2, every device or process, known as an island, work independently. The only communication between islands is when information from other processes is needed. Thus, a communication topology must be defined, it can be static or dynamic. For the sake of explanation, we consider a static communication topology.

In Algorithm 5, the communication between connected islands is shown. Each island initiates an independent EA and every few generations, a set of the best candidates are sent from island i to island j, for all $i \neq j$, and inserted into the population of the connected islands in order to share information.

The communication is not as time-consuming as the previous paradigm, thus, the communication overhead is reduced. Since every island is a different EA, the convergence of the algorithm may take longer, due to the perturbation of the algorithm introduced

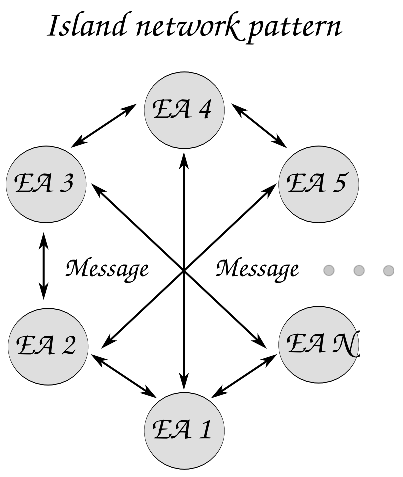


Figure 4.2: Schematic of an island topology network.

Algorithm 5 All-connected island network for evolutionary algorithms pseudocode

ISLAND$_i$

```
 1: Set $n \leftarrow 0$.
 2: Initialize algorithm parameters
 3: while stopCrit $\neq$ true do
 4:     Simulate a new population
 5:     if $n = NumGen$ then
 6:         Send best to $k = 1, \dots, S \; \forall \; k \neq i$
 7:         Receive from $k = 1, \dots, S \; \forall \; k \neq i$
 8:         Insert best into population
 9:         $n \leftarrow 0$
10:     else
11:         $n \leftarrow n + 1$
12:     end if
13:     Compute algorithm parameters
14: end while
```

ISLAND$_j$

```
 1: Set $n \leftarrow 0$.
 2: Initialize algorithm parameters
 3: while stopCrit $\neq$ true do
 4:     Simulate a new population
 5:     if $n = NumGen$ then
 6:         Send best to $k = 1, \dots, S \; \forall \; k \neq j$
 7:         Receive from $k = 1, \dots, S \; \forall \; k \neq j$
 8:         Insert best into population
 9:         $n \leftarrow 0$
10:     else
11:         $n \leftarrow n + 1$
12:     end if
13:     Compute algorithm parameters
14: end while
```

by the candidate solutions received from other islands. This can be seen as a benefit because of it can decrease the population bias towards local minima.

Chapter 5

Objective functions

To qualify an *"optimized mechanical configuration"*, a performance qualification must be applied to it. In this work, we refer as optimization problems the description of the proposed development problem to be solved. The mathematical expression that quantifies the performance based on the optimization problem is referred as an objective function. In the addressed optimization problems, we review objective functions in literature that allows quantifying the performance of proposed mechanisms. In this chapter, we review four models for optimum design, based on the classification of the objective functions introduced in Section 1.2. Three classes of problems are introduced as follows:

- **Static optimization problem**:
  1. *Minimization of the links lengths subject to reach given points in a workspace.* The user provides a set of points and the problem is to find the lengths of the mechanism which reach all the given points. The sum of the link lengths is **minimum**. Notice that minimizing the links length could be equivalent to minimize the material cost, power consumption and space used by the manipulator.
  2. *Maximization of a regular workspace subject to a constant norm of the links lengths.* The goal is to cover the **maximum** volume of a regular workspace, usually set as a cube, sphere or cylinder [11, 13, 14].
- **Kinematic optimization problem**:
  1. *Minimization of the trajectory tracking error for a concurrent optimal design.* Given a trajectory as a set of time-dependent points, the problem is to find the set of link lengths and control parameters that minimize the control signal,

defined as the weighted sum of the integrals of the norm of the position and velocities errors.

- **Dynamic optimization problem**:

  1. *Simultaneous minimization of the trajectory tracking error and applied torque for a concurrent optimal design.* As in the previous objective function, given a trajectory as a set of time-dependent points, the set of link lengths and control parameters are optimized to provide the minimum of the weighted sum of the integral of the absolute errors, computed as the difference of the desired and obtained values for each joint, and the sum of the integral of the torque applied at each joint [23–25].

Nevertheless, the same optimization algorithms and kinematic simulators can be used for approaching any of the optimization problems, the adequate model can be selected according to the application and user needs.

5.1 Objective 1: Minimum lengths subject to fulfill a workspace

A workspace W, possibly irregular, is discretized by a set of points $\mathbf{X} \in \mathbb{R}^{d\times n}$. The optimization problem is to find the minimal link lengths for reaching all the points $\mathbf{X} \in W$, taking into account joint limits, which specify a working range of joints angles. Notice that minimizing the lengths is equivalent to minimize the volume of the mechanism links. By instance, consider a mechanism which consists of q links, each link can be seen as a cylinder with a radius R_k, the volume of each link is denoted as $V_k = l_k(\boldsymbol{\alpha})R_k$, $k = 1, \dots, q$, where $l_k(\boldsymbol{\alpha})$ is the length of the link k. The total volume of the mechanism is computed as $\sum_{k=1}^{q} \pi l_k(\boldsymbol{\alpha})R_k^2$. Considering that all links have the same radius $R = R_k$, πR^2 is a constant and consequently, minimizing the volume of the mechanism is equivalent to minimizing the sum of the link lengths, subject to reaching the given points. This can be expressed as follows:

$$\min_{\boldsymbol{\alpha}} \mathcal{F}(\boldsymbol{\alpha}) = \sum_{k=1}^{q} l_k(\boldsymbol{\alpha}) + \lambda(1 - w(\boldsymbol{\alpha})), \tag{5.1}$$

subject to

$$\theta_i^{\min} \leq \theta_i \leq \theta_i^{\max}, \tag{5.2}$$

$$\phi_j^{\min} \leq \phi_j \leq \phi_j^{\max} \tag{5.3}$$

where $i = 1, \ldots, a$ and $j = 1, \ldots, m - a$. In Equation (5.1) $w(\boldsymbol{\alpha})$ is the percentage of reached points and $\lambda > 0$ is a weight which penalizes the objective function when the mechanism do not reach all the points. Notice that a point $\mathbf{X}_k$ is reachable by the mechanism if the inverse kinematic solution for that point exists and is within the joint limits in Equations (5.2) and (5.3). In this case and in the subsequent stated optimization problems, we assume that the joint limits are set in such a way that self-collisions are avoided, i.e., the working ranges of the joint angles are such that mechanical interference between links is not possible.

5.2 Objective 2: Maximum regular workspace

This problem consists in finding the maximum regular workspace for a mechanism whose sum of the links lengths is normalized to the unity [14]. The problem is subject to constraints that guarantee an adequate performance, such as manipulability constraints, which aim for keeping the robot away from singular configurations. A dexterity measure used in this work, which is often referred as an intuitive quantitative measure, is the inverse of the condition number of the Jacobian matrix $\kappa(\mathbf{J})$, defined as $\kappa(\mathbf{J}) = \sigma_{\min}(\mathbf{J})/\sigma_{\max}(\mathbf{J})$ [29], where $\sigma_{\min}(\mathbf{J})$ and $\sigma_{\max}(\mathbf{J})$ are the minimum and maximum singular values of the Jacobian matrix, respectively. Notice that, $\kappa \in [0, 1]$. From now on, we refer as manipulability to this dexterity measure introduced previously. In order to decouple translational and rotational manipulability of the end-effector, we can rewrite Equation (3.3) as follows:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{x}}_t \\ \dot{\mathbf{x}}_r \end{bmatrix} = \begin{bmatrix} \mathbf{J}_t \\ \mathbf{J}_r \end{bmatrix} \dot{\boldsymbol{\theta}} \tag{5.4}$$

where $\dot{\mathbf{x}}_t$ and $\dot{\mathbf{x}}_r$ are translational and rotational velocities of the end-effector, respectively. Thus, we can compute separately manipulability measures for position $\kappa(\mathbf{J}_t)$ and orientation $\kappa(\mathbf{J}_r)$. To guarantee position and orientation manipulability, constraints are imposed for each of these measures. Let us define a measure $\Phi(\boldsymbol{\alpha})$ for the volume of the workspace W. The measure $\Phi(\boldsymbol{\alpha})$ is expressed in terms of a parameter of the regular workspace, for instance, side length of a cube or radius of a sphere. Considering all the statements above, the optimization problem is defined as follows:

$$\max_{\boldsymbol{\alpha}} \mathcal{F}(\boldsymbol{\alpha}) = \Phi(\boldsymbol{\alpha}), \tag{5.5}$$

subject to

$$\begin{aligned}\kappa(\mathbf{J}_t(\mathbf{X},\theta,\boldsymbol{\alpha})) &\geq \gamma_1,\\ \kappa(\mathbf{J}_r(\mathbf{X},\theta,\boldsymbol{\alpha})) &\geq \gamma_2,\\ \theta_i^{\min} \leq \theta_i &\leq \theta_i^{\max},\\ \phi_j^{\min} \leq \phi_j &\leq \phi_j^{\max},\\ \sum_{k=1}^{q} l_k(\boldsymbol{\alpha}) &= \tau\end{aligned} \tag{5.6}$$

where $i = 1, \dots, a$, $j = 1, \dots, m-a$, γ_1 and γ_2 are position and orientation manipulability bounds, respectively. τ is a given normalizing constant, set to $\tau = 1$ in this work. The last constraint removes the dimension effects by normalizing the design parameters $\boldsymbol{\alpha}$ of the manipulator. Similar to the previous optimization problem, joint limits are specified for actuated and non-actuated joint variables. Additionally, we assume that joint limits are set adequately to avoid self-collisions. In our case study, we maximize the side length of a discretized cube covered by the workspace, which is equivalent to maximize its volume.

5.3 Objective 3: Concurrent minimization of joint error with a kinematic control

In this problem, we consider an actuated mechanism under a proportional-derivative control. Given a position and orientation function dependent on time, we aim to minimize the error between the desired trajectory and the actual trajectory follow by the mechanism. The error is a time-dependent function. Hence, the optimization problem considers the integral of the absolute value of the control signal, as follows:

$$\min_{\boldsymbol{\alpha}} \mathcal{F}(\boldsymbol{\alpha}) = k_p(\boldsymbol{\alpha}) \int_{t_0}^{t_n} ||\mathbf{e}(\boldsymbol{\alpha},t)||\partial t + k_d(\boldsymbol{\alpha}) \int_{t_0}^{t_n} ||\dot{\mathbf{e}}(\boldsymbol{\alpha},t)||\partial t, \tag{5.7}$$

subject to

$$\begin{aligned}\theta_i^{\min} \leq \theta_i &\leq \theta_i^{\max},\\ \phi_j^{\min} \leq \phi_j &\leq \phi_j^{\max}\end{aligned}$$

where $i = 1, \dots, a$, $j = 1, \dots, m-a$, $k_p(\boldsymbol{\alpha})$ and $k_d(\boldsymbol{\alpha})$ are proportional and derivative control gains, respectively, which are included as optimization variables in $\boldsymbol{\alpha}$. The functions $\mathbf{e}(\boldsymbol{\alpha},t) = \mathbf{X}(\boldsymbol{\alpha},t) - \mathbf{X}^D(t) \in \mathbb{R}^d$ and $\dot{\mathbf{e}}(\boldsymbol{\alpha},t) = \dot{\mathbf{X}}(\boldsymbol{\alpha},t) - \dot{\mathbf{X}}^D(t) \in \mathbb{R}^d$ are translation and velocity errors, respectively. $\mathbf{X}(\boldsymbol{\alpha}, t)$ and $\dot{\mathbf{X}}(\boldsymbol{\alpha}, t)$ are the translation and velocity coordinates (including orientation in both cases) of the manipulator at an instant t. Likewise, $\mathbf{X}^D(t)$ and $\dot{\mathbf{X}}^D(t)$ are the desired translation and velocity coordinates imposed to the manipulator by the desired trajectory. We consider that joint limits are specified for actuated and passive joint variables in such a way that self-collisions of the links are avoided.

The kinematic proportional-derivative control that allows the manipulator to track the desired trajectory is introduced as follows:

$$\dot{\boldsymbol{\theta}}_{t+1} = \mathbf{J}_t^{\dagger}(\dot{\mathbf{X}}^D(t) - k_P \mathbf{e}_t - k_D \dot{\mathbf{e}}_t) \tag{5.8}$$

where $\mathbf{J}_t^{\dagger} = \mathbf{J}_t^{\top}(\mathbf{J}_t \mathbf{J}_t^{\top})^{-1} \in \mathbb{R}^{a \times d}$ is the right Moore-Penrose pseudo-inverse of $\mathbf{J}_t$. Notice that if the number of actuated joints is equal to the number of DOF of the end-effector ($a = d$) then $\mathbf{J}_t^{\dagger} = \mathbf{J}_t^{-1}$. The values of the joints variables in the next instant of time are obtained using the Euler method, so that $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \Delta t \cdot \dot{\boldsymbol{\theta}}_{t+1}$. The translation and orientation variables in the next instant of time can be computed by using the Cartesian velocity in the current instant of time as follows: $\mathbf{X}_{t+1} = \mathbf{X}_t + \Delta t\ \dot{\mathbf{X}}_{t+1}$, where $\dot{\mathbf{X}}_{t+1} = \mathbf{J}_t \dot{\boldsymbol{\theta}}_{t+1}$.

5.4 Objective 4: Simultaneous optimization of joint error and consumed energy with a dynamic control

In this case, given a discretized trajectory, the objective function is taken as the weighted sum of two independent functions. The first is the sum of the integrals of the errors between the desired value and the position of each joint and the second is the integral of the torque applied by each joint, in both cases over time. This is expressed as:

$$\min_{\boldsymbol{\alpha}} \mathcal{F}(\boldsymbol{\alpha}) = \omega_1 \sum_{j=1}^{a} \int_{t=t_0}^{t_n} |e^{(j)}(t)| \partial t + \omega_2 \sum_{j=1}^{a} \int_{t=t_0}^{t_n} |\Delta \tau^{(j)}(t)| \partial t, \tag{5.9}$$

subject to

$$\theta_i^{\min} \leq \theta_i \leq \theta_i^{\max},$$
$$\phi_j^{\min} \leq \phi_j \leq \phi_j^{\max}$$

where ω_1 and ω_2 are the weights of each independent function. In this case, the position and velocity errors are given as $\mathbf{e}(\boldsymbol{\alpha}, t) = \boldsymbol{\theta}(\boldsymbol{\alpha}, t) - \boldsymbol{\theta}^D(t) \in \mathbb{R}^a$, $\dot{\mathbf{e}}(\boldsymbol{\alpha}, t) = \dot{\boldsymbol{\theta}}(\boldsymbol{\alpha}, t) - \dot{\boldsymbol{\theta}}^D(t) \in \mathbb{R}^a$. The difference of torque applied by the joint j each step of time is given as $\Delta\tau^{(j)}(t) = \tau^{(j)}(t) - \tau^{(j)}(t-1) \in \mathbb{R}^a$. This is a measure of the applied energy to move each link. In this case, a dynamic model with a multivariable PID controller is used to control the position and velocity of each link, by computing the torque applied to the mechanism, as shown in Appendix C.1. The simultaneous minimization of both functions drives the mechanism to track the desired trajectory as accurate as possible with the least amount of energy consumption. How accurate the mechanism is able to track the trajectory or how much energy consums are directly related to the weights and the value of the objective function in Eq. (5.9).

The weights applied to each independent function, scales and assigns the importance of each term in the objective function if the weights are known. Otherwise, we recommended the use of a multi-objective algorithm, where the user has to decide which configuration of functions fit better. For comparison reasons, we used the objective function as shown above with particular values of the weights. Below, two different versions of the objective function are presented for different weights.

5.4.1 Objective 4.1: Minimal joint error

Setting $\omega_1 = 1$ and $\omega_2 = 0$, the second independent function disappears from the expression in Eq. (5.9). Considering a discretized trajectory, a numerical integration algorithm is applied to compute the remaining term, which results in a weighted sum of the values of the function in terms of the discretized time. In this case, the objective function is to minimize the accumulated absolute error of each joint of the mechanism with respect to the desired trajectory, i.e.:

$$\min_{\boldsymbol{\alpha}} \mathcal{F}(\boldsymbol{\alpha}) = \sum_{j=1}^{a} \sum_{t=t_0}^{t_n} |e^{(j)}(t)|, \tag{5.10}$$

subject to

$$\theta_i^{\min} \leq \theta_i \leq \theta_i^{\max},$$
$$\phi_j^{\min} \leq \phi_j \leq \phi_j^{\max}.$$

The minimization of the cumulative absolute error drives the mechanism to track the desired trajectory as accurate as possible. It is worth to mention that the value of the objective function can change if the step time of the discretization changes, since a different number of values of the error are introduced in the sum. Thus, we suggest to solve the integral of the error function instead of summing its values in order to normalize with respect to time. Nevertheless, we use the objective function as presented to compare the results with the ones found in literature.

5.4.2 Objective 4.2: Minimal energy consumption

In this case, setting $\omega_1 = 0$ and $\omega_2 = 1$, the first term of the function in Eq. (5.9) is not considered. If the trajectory is a discretized function of time, we have to perform a numerical integration. Without considering weighting by the time step (normalization by time step), we have:

$$\min_{\boldsymbol{\alpha}} \mathcal{F}(\boldsymbol{\alpha}) = \sum_{j=1}^{a} \sum_{t=t_0}^{t_n} |\Delta\tau^{(j)}(t)|, \tag{5.11}$$

subject to

$$\theta_i^{\min} \leq \theta_i \leq \theta_i^{\max},$$
$$\phi_j^{\min} \leq \phi_j \leq \phi_j^{\max}.$$

This minimization of the cumulative absolute difference of torque drives the mechanism to the desired trajectory by using the minimum amount of energy by each link. As in the previous objective function, the variation of the time step delivers different results. Thus, we suggest using the integration of the cumulative absolute torque difference with respect to time. However, for the sake of comparison, we use the objective function as presented without time step normalization.

Chapter 6

Proposed Methodology

Once we have introduced the optimization problems and the tools to be used for solving them, we have the elements to present, in this chapter, the proposed methodology to approximate the solution of the concurrent optimization design of kinematically complex mechanisms. We also introduce a new parallelization proposal for an EDA. It is based on a mathematical derivation of formulas for estimating parameters, in the master process, of a global probability distribution, which concentrates the information of local probability distributions from the slave processes. Finally, the parallel algorithm is used to approximate optimal solutions of several optimization problems introduced in the last chapter.

6.1 Proposed optimization methodology

Let us describe the proposed general methodology for optimization of a kinematically complex mechanism. This methodology can be used for static, kinematic or dynamic optimization problems, in the last two cases, we consider a concurrent optimization problem: the optimization of geometry and control parameters simultaneously subject to a certain task.

Fig. 6.1 shows a graphical representation of the methodology, which summarizes the process in three main steps:

1. Firstly, the mechanism to be optimized must be selected together with a set of design parameters, e.g., masses, fixed lengths, inertial moments, desired workspace size and shape, etc., as well as a kinematic or dynamic model.

2. Secondly, the objective function must be selected consequently, a control method, trajectory or workspace must be selected in agreement with the optimization model.

3. Lastly, an optimization algorithm must be selected, from the three options described in Chapter 4, and the parameters of such algorithm must be inputted.

These three steps provide a method whose output is a set of design parameters (geometric and control parameters) that approximates the optimal solution.

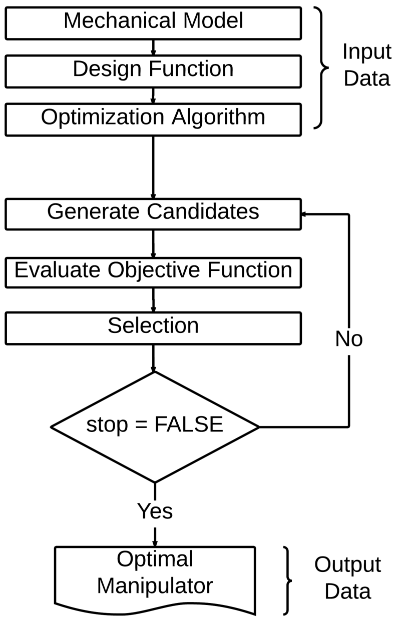


FIGURE 6.1: Graphic representation of the concurrent optimization method for kinematically complex mechanisms.

As can be seen, this methodology could be used to optimize any mechanism under the requirement of providing an evaluator of a function which describes its quality.

6.2 Proposal of parallelization

Firstly, let us recall Section 4.2, where the most commonly used parallelization paradigms are presented. In that section, we introduce the proposed parallelization paradigm of an EDA. We aim to take the advantages of both previous paradigms and overcome their main disadvantages, e.g., overhead because of communication. The Algorithm 6 shows the communication protocol for the proposed parallelization. In the parallel algorithm, the master process initializes the parameters of the probability distribution, while the slave processes randomly their parameters. The main difference with the classic master-slave paradigm is that instead of sending a set of candidate solutions to every slave to be evaluated, the estimated parameters are sent. Once the slaves have received the estimation parameters, a subpopulation is generated. Then, the new population is evaluated and new parameters are computed and sent back to the master node.

Algorithm 6 Proposed Parallelization of an Estimation Distribution Algorithms Pseudocode

```
MASTER
 1: Initialize estimation parameter: $\lambda_0$
 2: Set $t \leftarrow 0$
 3: while stopCrit $\neq$ true do
 4:     for $i = 1 : \mathcal{N}$ do
 5:         Send to SLAVE$_i$ : $\lambda_t$
 6:     end for
 7:     for $i = 1 : \mathcal{N}$ do
 8:         Receive from SLAVE$_i$ : $\lambda^{(i)}_{t+1}$
 9:     end for
10:     Weighting estimation parameter: $\overline{\lambda}_{t+1}$
11:     $t \leftarrow t + 1$
12: end while
```

```
SLAVE$_i$
1: Initialize pseudo-random parameters
2: while stopCrit $\neq$ true do
3:     Receive from MASTER : $\lambda_t$
4:     Simulate subpopulation
5:     Evaluate subpopulation
6:     Compute $\lambda^{(i)}_{t+1}$
7:     Send to MASTER : $\lambda^{(i)}_{t+1}$
8: end while
```

The master considers the information from all processes by the weighted sum of the estimated parameters from each subpopulation. Finally, the random seed and fitness of the best candidate from every slave are sent to the master, so that the best candidate from each subpopulation can be generated, if needed. The proposed parallelization reduces the memory usage of the messages.

Let us recall Eq. (4.2), where the total transmission time of a standard master-slave communication protocol is computed as $t_{total} = t_f\phi_p + S\left(\beta\phi_p l_i + L\right) + \left(\beta\phi_p l_f + L\right)$. We can divide the data transmission routine into two, the subpopulation transmission routine n, where a set of candidate solutions are sent from the master to the i^{th} slave and the function evaluations are sent back to master, and the processes transmission routine m, where the previous routine is executed S times in order to distribute the population. As can be seen, there is a $n \times m$ data transmissions between processes, assuming that $m = n$, the complexity of the algorithm is $O(n^2)$. As for the proposed parallelization, the total transmission time between processes can be computed as:

$$t_{total} = t_f\phi_p + (S+1)\left(\beta l_{\lambda_i} + L\right) + \left(\beta l_{\mathcal{F}} + L\right) \tag{6.1}$$

where S is the number of slave processes, l_{λ_i} and $l_{\mathcal{F}}$ are the length of the local distribution parameters and the evaluation of the function in bytes, respectively. In this case, the only transmission routine is the distribution parameters transmission routine n, where the distribution parameters are sent between master and slaves, therefore, there is a n data transmissions between processes and the complexity of the algorithm is $O(n)$.

The performance of the method lies in the reconstruction of a global probability distribution model that relates the probability distributions from the slave processes. The weighting in the parameter estimation plays an important role in the proposed algorithm, it is described in the next section.

6.2.1 Weighting in the formulae for parameter estimation

Let us introduce two types of weighting methods to compute the global probability distribution. Firstly, let us assume that the global probability distribution used to sample candidate solutions is denoted by $Q_0(\lambda_0, x)$, where λ_0 its parameters. The probability distribution computed by each i^{th} slave is denoted by $Q_i(\lambda_i, x)$, where λ_i are the parameters of the i^{th} probability distribution. A general parameter weighting is given as:

$$\lambda_0 = \sum_{i=1}^{S} \omega_i \lambda_i \tag{6.2}$$

where ω_i is the weight for each local distribution parameter. The first proposed weighting method is an average-based estimation parameter. Each weight is computed as $\omega_1 = \ldots = \omega_S = 1/S$. In Fig. 6.2, we graphically represent the behavior of the first weighting method, described above.

In this experiment, several bivariate Gaussian distributions are drawn randomly, represented as blue ellipses with a 0.95 of confidence, the global bivariate Gaussian distribution is represented in red (see Fig. 6.2). In the first row of the figure, the local covariance matrixes are equally fixed and the local mean vectors are generated randomly, the average method sets the global mean vector on the centroid of the distributions. In the second row, the local mean vectors are equally fixed and the local covariance matrixes are generated randomly, where the global covariance matrix it is oriented to the average of directions. In the last row of the figure, the local mean vectors and the local covariance matrixes are randomly generated. In this case, the global distribution tends to orient and place the candidate solution at the centroid of the local distributions. This weighting method does not generate a robust estimator since it does not take into account the uncertainty of each local distribution and values of the objective function of the local populations.

The second weighting method is derived from a minimization problem. The parameters are computed by minimizing the sum of the Kullback-Leibler divergence between the

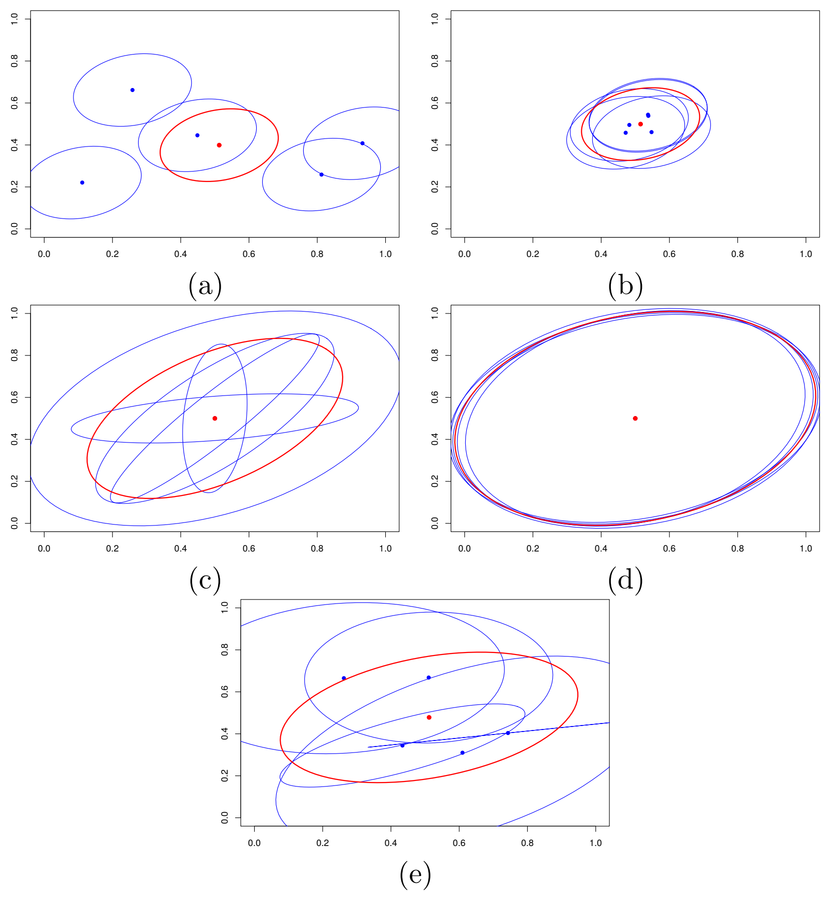


FIGURE 6.2: Visualization of behaviour of the average weighting using multivariate Gaussian distributions. (a) Wide randomized mean vector. (b) Near randomized mean vector. (c) Wide randomized covariance matrix. (d) Near randomized covariance matrix. (e) Average weighting.

global probability distribution and the probability distribution from each i^{th} subpopulation, as shown in Eq. (6.3).

$$\min_{\lambda_0} \sum_{i=1}^{N} D_{\mathcal{KL}} \left(Q_0 || Q_i \right), \tag{6.3}$$

with

$$D_{\mathcal{KL}} \left(Q_0 || Q_i \right) = \int_{-\infty}^{\infty} q_0(x) \log \frac{q_0(x)}{q_i(x)} dx \tag{6.4}$$

where $D_{\mathcal{KL}} \left(Q_0 || Q_i \right)$ is the Kullback-Leibler divergence between the global probability distribution and the i^{th} probability distributions. $q_0(x)$ and $q_i(x)$ are the probability density functions of the global and the i^{th} process, respectively.

Let us now assume that the EDA uses an univariate Gaussian distribution to sample new candidate solutions, thus, the Kullback-Leibler divergence is computed as:

$$D_{\mathcal{KL}}\left(Q_0 || Q_i\right) = \log \frac{\sigma_i}{\sigma_0} + \frac{\sigma_0^2 + (\mu_0 - \mu_i)^2}{2\sigma_i^2} - \frac{1}{2} \tag{6.5}$$

where μ_0 and μ_i are the means of the univariate Gaussian distribution of the global and the i^{th} distribution, respectively. σ_0 and σ_i are the standard deviation of the univariate Gaussian distribution of the global and the i^{th} distribution, respectively.

To minimize Eq. (6.3), it must be differentiated with respect to the global distribution parameters. Due to it is a sum, we differentiate each term independently. Thus, the derivative of each term of Eq. (6.5) are:

$$\frac{\partial D_{\mathcal{KL}}}{\partial \mu_0} = \frac{\mu_0 - \mu_i}{\sigma_i^2} \quad ; \quad \frac{\partial D_{\mathcal{KL}}}{\partial \sigma_0} = \frac{\sigma_0}{\sigma_i^2} - \frac{1}{\sigma_0}. \tag{6.6}$$

Using Eq. (6.6), we can compute the estimators by adding the contribution of each differentiation of Eq (6.5) and equating it to zero. As a result, we can compute the mean and variance estimators as follows:

$$\mu_0 = \left[\sum_{i=1}^{N} \frac{1}{\sigma_i^2}\right]^{-1} \sum_{i=1}^{N} \frac{\mu_i}{\sigma_i^2} \quad ; \quad \sigma_0^2 = N \left[\sum_{i=1}^{N} \frac{1}{\sigma_i^2}\right]^{-1}. \tag{6.7}$$

In the same vein, we can derive formulae to compute estimators of the parameters of a multivariate Gaussian distribution, which is used to sample new candidate solutions. The Kullback-Leibler divergence is computed as:

$$D_{\mathcal{KL}}\left(Q_0 || Q_i\right) = \frac{1}{2}\left[\log \frac{|\boldsymbol{\Sigma}_i|}{|\boldsymbol{\Sigma_0}|} - d + trace\left(\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\Sigma}_0\right) + \left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_i\right)^T \boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_i\right)\right] \tag{6.8}$$

where $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_i$ are the mean vectors of the multivariate Gaussian distribution, and $\boldsymbol{\Sigma}_0$ and $\boldsymbol{\Sigma}_i$ are the covariance matrices for the global and the i^{th} distribution, respectively. We distinguish four different terms in Eq. (6.8) , a logarithmic term, a constant dimensional term, a matrix trace term and a quadratic term.

To minimize Eq. (6.3), we must differentiate Eq. 6.8. To do so, we differentiate each term independently with respect to the parameters of the global probability. Let us first differentiate with respect to $\boldsymbol{\mu}_0$. As can be seen in Eq (6.8), the quadratic term in the expression is the only one in function of $\boldsymbol{\mu}_0$. Hence,

$$\frac{1}{2}\frac{\partial}{\partial \boldsymbol{\mu}_0}\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_i\right)^T \boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_i\right) = \boldsymbol{\Sigma}_i^{-1}\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_i\right). \tag{6.9}$$

Now, we must differentiate with respect to $\mathbf{\Sigma}_0$. As can be seen in Eq. (6.8), there are two terms that are in function of $\mathbf{\Sigma}_0$, the logarithmic and the matrix trace terms. Firstly, let us analyze the matrix trace term. Using the definition of the trace of a matrix multiplication, as $trace(\mathbf{C}) = \sum_{i=1}^{d} c_{i,i}$, where $\mathbf{C}$ is the multiplication of matrices $\mathbf{A} \in \mathbb{R}^{d\times d}$ and $\mathbf{B} \in \mathbb{R}^{d\times d}$, if $trace(C)$ is differentiated with respect to $\mathbf{A}$, for $a_{m,n}$:

$$\frac{\partial c_{i,i}}{\partial a_{m,n}} = \frac{\sum_i \left[\sum_j \partial(a_{i,j} b_{j,i})\right]}{\partial a_{m,n}} \tag{6.10}$$

where a and b are the elements of the matrices $\mathbf{A}$ and $\mathbf{B}$, respectively. For $1 \leq n, m, i, j \leq d$, if $i = m$ and $j = n$, $\frac{\partial c_{i,i}}{\partial a_{m,n}} = b_{n,m}$, otherwise, the differentiation is 0, thus, we obtain $\frac{\partial \mathbf{C}}{\partial \mathbf{A}} = \mathbf{B}^T$. Let us now analyze the logarithmic term:

$$\frac{\partial \log |\mathbf{\Sigma}_0|}{\partial \mathbf{\Sigma}_0} = \frac{\partial |\mathbf{\Sigma}_0| / \partial \mathbf{\Sigma}_0}{|\mathbf{\Sigma}_0|}. \tag{6.11}$$

Once the determinant of $\mathbf{\Sigma_0}$ is differentiated, we can see a correlation between the inverse matrix, the inverse of the determinant and the adjoint matrix of the global covariance matrix. Thus, $\partial|\mathbf{\Sigma}_0|/\partial\mathbf{\Sigma}_0 = |\mathbf{\Sigma}_0|\mathbf{\Sigma}_0^{-1^T}$, as shown in [45]. Eq. (6.12) shows the final differentiation of the Eq. (6.8) with respect to the parameters of the global probability.

$$\frac{\partial D_{\mathcal{KL}}}{\partial \boldsymbol{\mu}_0} = \mathbf{\Sigma}_i^{-1}\left(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_i\right) \quad ; \quad \frac{\partial D_{\mathcal{KL}}}{\partial \mathbf{\Sigma}_0} = \frac{1}{2}\left[\left(\mathbf{\Sigma}_i^{-1}\right)^T - \mathbf{\Sigma}_0^{-1^T}\right]. \tag{6.12}$$

Adding the contributions of each differentiation and equating the result to zero. After an algebraic manipulation of the previous expressions, we can compute the estimators of the parameters as follows:

$$\boldsymbol{\mu}_0 = \left[\sum_{i=1}^{N} \mathbf{\Sigma}_i^{-1}\right]^{-1} \sum_{i=1}^{N} \mathbf{\Sigma}_i^{-1} \boldsymbol{\mu}_i \quad ; \quad \mathbf{\Sigma}_0 = N\left[\sum_{i=1}^{N} \left[\mathbf{\Sigma}_i^{-1}\right]\right]^{-1}. \tag{6.13}$$

In both cases, if we assume that in every i^{th} probability distribution, the covariance matrices are the same, it results in the average-based weighting method.

We present different visual experiments to see the behavior of this weighting method, shown on Fig 6.3. In these experiments, several Gaussian distributions equidistantly spaced are generated. The main idea is to see how the mean vector of the global distribution is attracted due to the covariance matrix of each i^{th} Gaussian distributions and resulting global covariance matrix.

As can be seen in Fig. 6.3, the global covariance matrix and the global mean vector are highly related to the local covariance matrices. The local covariance matrices act as

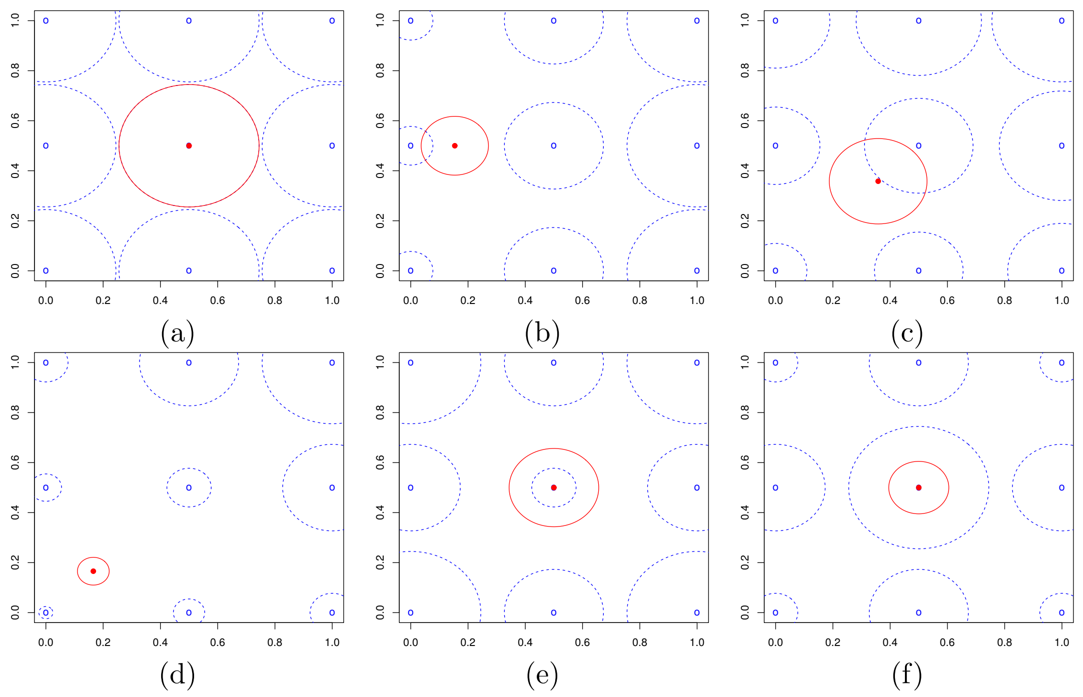


FIGURE 6.3: Visualization of behaviour of the variance weighting using multivariate Gaussian distributions. (a) Same covariance matrix. (b) Linear attractor. (c) Light corner-attractor. (d) Heavy corner-attractor. (e) Light corner-attractors. (f) Heavy corner-attractors.

attractors to the global distribution. The smaller the covariance matrix is, the higher impact it has in the estimation formula.

6.3 Concurrent Optimization Design for Mechanisms (*CODeMe*) Software

In this section, the implemented software to solve the concurrent optimization design of kinematically complex mechanisms is described. This software follows the methodology described in Section 6.1. Concurrent Optimization Design for Mechanisms (*CODeMe*) is an open source software, that allows the user to select from a range of optimization problems, kinematic and dynamic models and optimization methods. It also allows the user to code their own mechanical model and objective functions. As in the methodology presented above, firstly, the user must select the kinematic or dynamic model of the mechanism, secondly, a objective function is chosen, and thirdly the optimization method to approach the problem must be selected.

CODeMe is coded in C, consisting of 3 different programming levels, each level supports the higher levels. Let us briefly introduce them.

- **Level 0**. Consist of libraries for a pseudo-random number generator, matrix and vector operations and functions for pipe-based process communications, which provide support to higher levels.
- **Level 1**. Consist in the implementation of the kinematic and dynamic models, and the kinematic and dynamic control to simulate the mechanisms. The mathematical models and control schemes implemented on *CODeMe* are described in Appendix A, B and C. A brief description of the implemented models are presented as follows:
  - *Forward/Inverse kinematics.* A general forward kinematic model for serial manipulator is implemented as presented in Appendix A.1.1. The particular implementation of the anthropomorphic serial manipulator and the two-link planar manipulator are presented in appendixes A.1.3 and A.1.4 respectively. The inverse kinematics for the parallel Delta manipulator is implemented as presented in Appendix B.1.1.
  - *Differential kinematics models.* The forward and inverse differential kinematics are implemented based on the solution of the system of equations presented in Section 3.2. A general algorithm to compute the Jacobian matrix for a serial manipulator is implemented as presented in Appendix A.1.2. The particular implementation for the anthropomorphic serial manipulator, the two-link planar manipulator and the parallel Delta manipulator are presented in Appendixes A.1.3, A.1.4 and B.1.2 respectively.
  - *Dynamic models.* The implementation receives the inertial matrix for each link and the inertial reference position of each link, which are used to compute the complete inertial matrix, the Coriolis matrix and the gravitational vector. The forward and inverse dynamics are implemented based on the solution of the system presented in Section 3.3. A general dynamic model for serial manipulators is implemented as presented in Appendix A.2. The particular dynamic implementations for the two-link 2-DOF planar manipulator and the counterweighted two-link 2-DOF planar manipulator were carried out, as presented in Appendixes A.2.1 and A.2.2 respectively.
- **Level 2**. In this last level, the optimization methods and objective functions are implemented. The optimization methods used are the ones presented in Chapter 4 and the source codes are provided on the web page of the corresponding authors. The implemented objective functions are presented in Chapter 5.

CODeMe can be downloaded from the website: http://www.cimat.mx/ ivvan/public/-codeme.html.

Chapter 7

Results on case studies

In this chapter, three case studies are presented, for which we address the optimization problems stated in Chapter 5 using the optimization algorithms described in Chapter 4. The first case of study is an anthropomorphic manipulator of 6-DOF, the second one is a 3-DOF Delta parallel manipulator and finally a 2-DOF two-link planar manipulator.

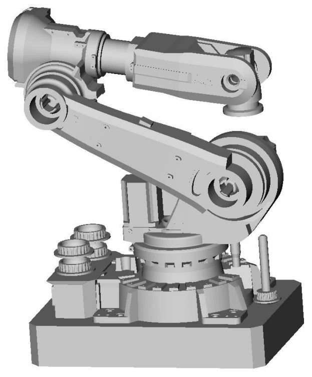
We report results and discussions of the comparison for the different optimization algorithms. We have used an intensive sampling technique in order to conclude which algorithm delivers the best performance value of the design problem.

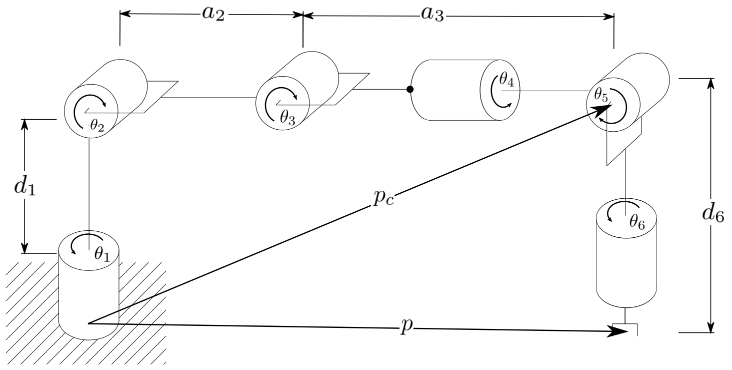
Each optimization method is executed 10 times for each design problem to present a statistical comparison. The stopping criteria are the following: the CMA-ES stops if the best objective function value is less than $1.0e$-9, or the difference between two consecutive generations is less than $1.0e$-20. The Omni-optimizer stops when it reaches 40 generations. The BUMDA stops if the maximum variance is less than $1.0e$-12. All algorithms stops at a maximum of $1e4{\cdot}p$ evaluations, for p optimization variables.

7.1 Case study: Anthropomorphic serial manipulator

Serial robotic manipulators are open-loop kinematic chains that consisting of interconnected joints and links. Because of their architecture, there is a general close-form equation for the forward kinematics and a $6 \times n$ Jacobian matrix, as presented in Appendix A.

An anthropomorphic manipulator, as shown in Fig. 7.1(a), is a 6-DOF serial manipulator interconnected through actuated revolute joints. Fig. 7.1(b) shows a representation of the six DOF of the manipulator, a detailed description of the kinematics of this mechanism are presented in Appendix A.1.3.


(a)



(b)

FIGURE 7.1: Case study: Anthropomorphic serial manipulator. (a) Schematic of the architecture of an anthropomorphic robot [2]. (b) Serial chain of an anthropomorphic robot.

7.1.1 Minimum lengths mechanism design subject to fulfill a workspace

The first optimization problem for the anthropomorphic manipulator is defined in Section 5.1. It consist in the minimization of the links lengths, subject to the reachability of the desired workspace. For this purpose, the base reference frame of the manipulator is set as the origin of the Cartesian space and a spherical workspace centered in $(200, 300, 500)$ with radius of 250 is discretized into 256 points, obtained randomly to define a cloud of points that form an irregular workspace. A gradient descent method is used to obtain the angular joints positions $\boldsymbol{\theta}$ (inverse kinematics) by minimizing the norm of the difference between the desired Cartesian position of the end-effector and an initially given position, this is $f(\mathbf{X}) = ||\mathbf{X}_D - \mathbf{X}||$. The solution is a configuration which minimizes $f(\mathbf{X})$ regardless of the orientation of the end-effector. Table 7.1 shows the results of the optimization obtained by the EAs, where d_1, a_2, a_3 and d_6 are the designed links lengths, corresponding to the best objective function value $\mathcal{F}(\boldsymbol{\alpha})$. The search range of the links lengths is $[0, 800]$ for lengths d_1, a_2, a_3 and d_6.

TABLE 7.1: Best solution for Eq. (5.1), links lengths $\{d_1, a_2, a_3, d_6\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$.

EA	d_1	a_2	a_3	d_6	$\mathcal{F}(\boldsymbol{\alpha})$
CMA-ES	$3.03e{+}2$	$3.66e{+}2$	$4.31e{+}2$	$5.26e{+}1$	$1.14e{+}3$
OMNI	$3.06e{+}2$	$3.67e{+}2$	$4.38e{+}2$	$5.31e{+}1$	$1.15e{+}3$
BUMDA	$3.02e{+}2$	$3.67e{+}2$	$4.33e{+}2$	$5.23e{+}1$	$1.15e{+}3$

In Table 7.1, the best optimal solutions found by each EA is presented. Note that the geometric parameters found by the three optimization method are near each other, meaning that the proposed geometric configuration of the CMA-ES, with the best performance, approximate to the optimal solution.

7.1.2 Optimal concurrent mechanism design with a kinematic control

The second optimization problem for the anthropomorphic manipulator is defined in Section 5.3. It has the objective to find the geometric and control parameters by minimizing the error of a tracked trajectory. The implementation receives a number of points distributed over time, which are used for interpolate the trajectory that is tracked by the end-effector. The desired velocities are computed by differentiating the interpolated trajectory, where $x(t)$, $y(t)$ and $z(t)$ are interpolated independently with respect to time, thus, C^2-curves are used in order to generate smooth position and velocity profiles. For this case of study, the forward kinematics is computed, thus, in contrast with the case in subsection 7.1.1, we do not need to solve an optimization problem to obtain the angles of a configuration. The target trajectory is a tridimensional spiral sampled in 100 equidistant points in the time dimension. The desired orientation for each point is computed from the corresponding direction cosines, i.e., the desired trajectory for the configuration of the end-effector is defined by $\mathbf{X}_i(t) = \langle c_1 \cos(0.5t), c_1 \sin(0.5t), (c_2 t + 0.2), \beta, \gamma, \eta \rangle$, where β , γ and η are the direction cosines of the corresponding point, for $c_1 = 0.3$, $c_2 = 0.1$, $0.0 \leq t \leq 30.0$ s and $i = 1, \ldots, 100$. A very large value proportional to the remaining simulation time is assigned to unfeasible mechanisms which can not be simulated due to singularities. The search range of the links lengths is $[0, 800]$ for lengths d_1, a_2, a_3 and d_6, and of the control parameters is $[0, 200]$ for gains k_p and k_d.

Table 7.2: Best solution for Eq. (5.7), links length $\{d_1, a_2, a_3, d_6\}$, control gains $\{k_p, k_d\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$.

EA	d_1	a_2	a_3	d_6	k_p	k_d	$\mathcal{F}(\boldsymbol{\alpha})$
			Proportional Control				
CMA-ES	3.00e+2	5.82e+2	6.30e+2	1.44e+2	1.06e+1	-	1.88e+3
OMNI	2.98e+2	1.95e+2	6.31e+2	1.40e+2	1.05e+1	-	1.88e+3
BUMDA	3.02e+2	5.67e2	6.31e+2	1.39e+2	1.06e+1	-	1.87e+3
			Proportional-Derivative Control				
CMA-ES	3.09e+2	5.71e+2	5.16e+2	8.49e+1	8.26e+1	8.30e-1	2.55e+3
OMNI	3.18e+2	5.62e+2	5.07e+2	6.46e+1	6.70e+1	8.38e-1	3.84e+3
BUMDA	3.17e+2	5.62e+2	5.17e+2	5.81e+1	7.93e+1	8.41e-1	3.85e+3

Table 7.2 shows the results obtained by the EAs. As noted, the solutions presented are near each other, possibly due to the proximity to the optimal solution. In this case, BUMDA and CMA-ES deliver the best solution for the proportional (P) and proportional-derivative (PD) control, respectively. Fig. 7.2 (a) and (c) show the logarithmic absolute End-Effector Position Error (EEPE) over time for the proportional (Eq. (5.7) with $k_D = 0$), and proportional-derivative control of the best found solution, respectively. Figs. 7.2 (b) and (d) show in green the followed path by the anthropomorphic

robot and the desired path in blue, for the proportional (P) and proportional-derivative (PD) control, respectively. It can be seen a fast and accurate tracking, where at 3.0 seconds, the order of magnitude of the error is $1e$-1. Showing that the mechanism can smoothly and accurately track the trajectory given by the points with both P and PD controllers.

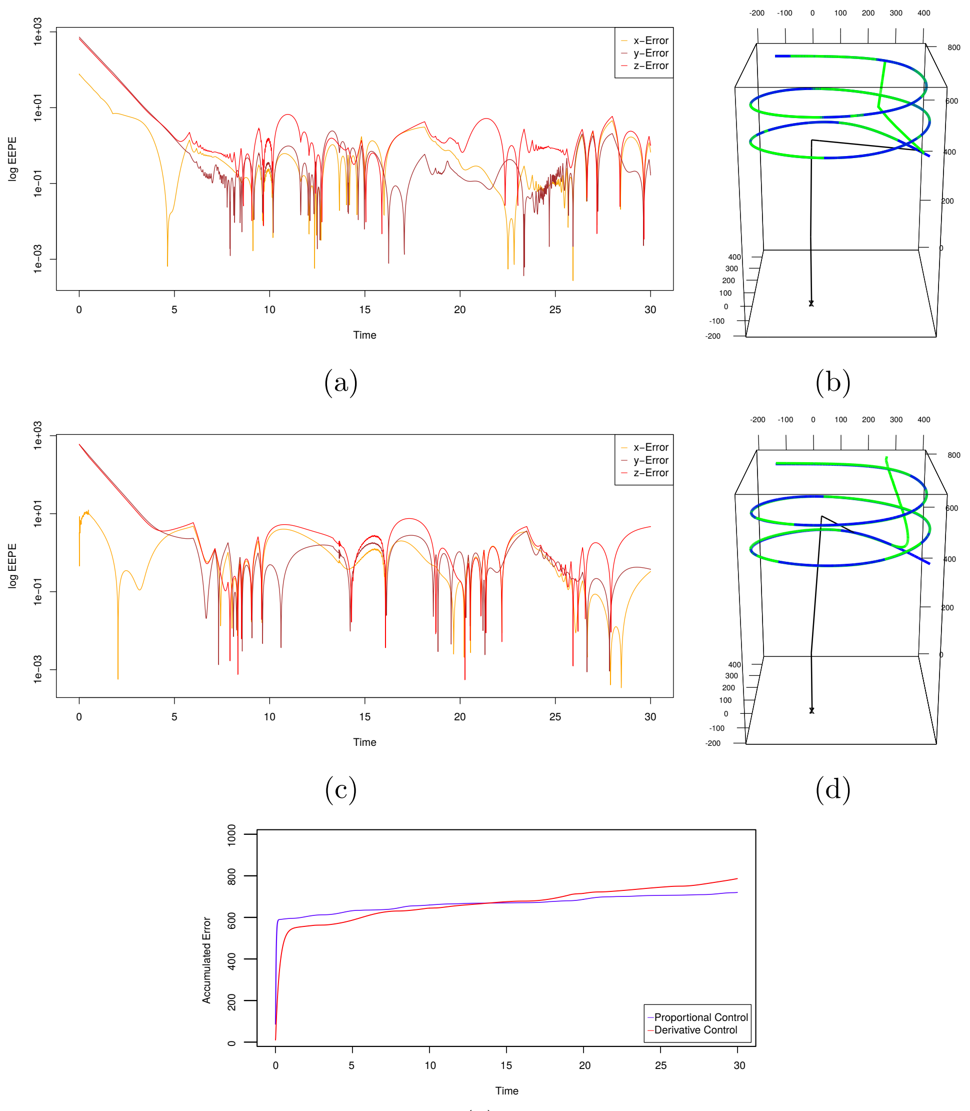


FIGURE 7.2: Trajectory tracking control for the anthropomorphic robot. (a) Logarithmic absolute EEPE over time for P control. (b) P path tracking. (c) Logarithmic absolute EEPE over time for PD control. (d) PD path tracking. (e) Absolute EEPE integral for P and PD control.

Fig. 7.2 (e) describes the integration of the control signal over time, in order to compare energy consumption of the P and PD controllers. As shown, the P control reaches faster the target trajectory at a higher energy consumption (high gain control). The optimal control gains of the PD control are smaller and it generates an initial motion of the robot that is actually feasible for a real one. Recall that the control parameter k_p and k_d are optimized at the same time as the geometric parameters $\boldsymbol{\alpha}$.

Table 7.3 shows a comparison between optimization methods presented in section 4, which are used to solve the concurrent optimization problem. The evolutionary algorithms are compared using a hypothesis test, where the null hypothesis (H_0) is that the mean value of the algorithm A (μ_A) is the same as the mean values of Algorithm B (μ_B). The alternative hypothesis (H_1) is that algorithm A is better than B. Because these algorithms are used in minimization problems, in this hypothesis H_1 is $\mu_A < \mu_B$, for the maximization case shown in the design problem presented in Section 5.2, the objective function values are multiplied by -1.

A similar hypothesis testing is implemented into the optimization variables to verify the accuracy of the evolutionary algorithms. For this hypothesis testing, the null hypothesis (H_0) is that the mean of the optimization variable values delivered by algorithm A, (α_A), is less than that delivered by algorithm B, (α_B), the alternative hypothesis (H_1) is that the function parameter (α_A) is bigger than the function parameter (α_B). After applying the hypothesis test, the p-values are computed. This value defines the probability of H_0 to be met. If the p-value is less than the significance level, it indicates that is most likely that H_1 is satisfied [46].

Each algorithm is executed 10 times for each optimization problem under the same conditions and an intensive sampling technique (bootstrap [47]) is applied to compare the algorithms.

TABLE 7.3: Results of the hypothesis test ($\mu_A < \mu_B$) with a 5% significancefor the optimization of an anthropomorphic serial manipulator, with the objective functions described in Eq. (5.1) and Eq. (5.7).

EA$_1$ vs EA$_2$	d_1	a_2	a_3	d_6	k_p	k_d	$\mathcal{F}$
1.- Optimal minimum lengths mechanism design subject to fulfill a workspace							
CMA-ES vs BUMDA	=	< (0.02204)	=	=	-	-	0.108 *
CMA-ES vs OMNI	< (0.0023)	=	< (0.00013)	=	-	-	0.000
BUMDA vs OMNI	> (0.0035)	=	> (0.00021)	=	-	-	4.000e-5
2.- Optimal concurrent mechanism design with a kinematic proportional control							
CMA-ES vs BUMDA	=	=	=	=	=	-	0.106 *
CMA-ES vs OMNI	=	=	=	< (0.00248)	=	-	4.800e-4
BUMDA vs OMNI	=	=	=	=	=	-	2.546e-2
3.- Optimal concurrent mechanism design with a kinematic proportional-derivative control							
CMA-ES vs BUMDA	< (0.00248)	=	=	=	=	=	1.259e-2
CMA-ES vs OMNI	=	=	=	=	=	=	0.182 *
BUMDA vs OMNI	=	=	=	=	=	=	0.828 *

Table 7.3 can be read as follows: For each optimization parameter the p-value is shown. If the mean of the optimized parameter of algorithm A is greater that the one of algorithm B, it is labeled with $>$. If the opposite occurs, the mean of the optimized parameter of algorithm A is smaller than that of algorithm B, it is labeled with $<$. The p-value for the corresponding hypothesis test is shown in parenthesis. If both algorithms have a similar mean of the optimization variable values, we use the label $=$. For instance, the second term of the first row of the table shows that the mean of the optimized parameter a_2 delivered by CMA-ES is smaller than the one delivered by BUMDA. The means of the other parameters are similar for both algorithms.

The p-value of the fitness function hypothesis test is shown in the last column, it is the proportion of times an algorithm is better than the other. If none of the algorithms is in bold, both algorithms have similar mean and reach similar optimal solutions. For instance, the second row of the table shows that the CMA-ES has a lower mean than Omni-optimizer with a p-value of 0.108. The same representation is used in the following sections when other algorithms are compared.

As seen in Table 7.3, the CMA-ES found the best optimal solutions more often than Omni-optimazer and BUMDA. In most of the cases, the optimal parameters of the CMA-ES and Omni-optimazer are similar. This gives statistical confidence to the result, due to both algorithms must have sampled many solutions in the vicinity of this value using different methodologies.

7.2 Case study: Delta parallel manipulator

In this section, we present the second case of study, which is a Delta parallel manipulator, considered as a kinematically complex mechanism according to our definition in Section 1.3. The Delta manipulator, as shown in Fig. 7.3, is a 3-DOF purely translational parallel robot. This mechanism is well-known for its high speed and accuracy [6]. The optimization problems, applied to the Delta parallel manipulator are addressed as follow.

7.2.1 Maximum regular workspace mechanism design

The first problem addressed for the Delta robot was defined in Section 5.2. We aim to find the maximum regular workspace within the total workspace of the mechanism. To this end, we define a cubic shape for a regular workspace W. The length of an edge of the cubic workspace is denoted by $2l$ and it is used to compute the objective function in Eq. (5.5).

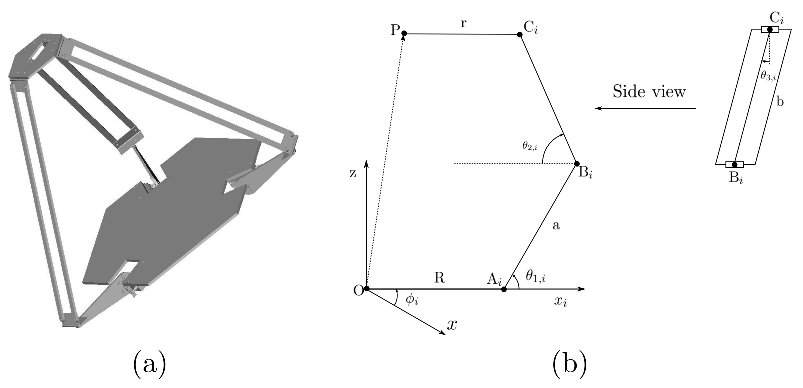


FIGURE 7.3: Second case of study: Delta parallel manipulator. (a) Schematic of the architecture of the Delta robot [3]. (b) The i^{th} subchain of the Delta robot.

The center of the resultant maximal effective regular workspace is unknown for the evaluation of each candidate solution, but it is known that the manipulator is symmetrical with respect to the axis z. In consequence, the coordinates of the center of the workspace has the form $(0, 0, z_c)$. In our work z_c is found by intensive search discretizing the axis in partitions of size $1.0e$-5.

For this case study, the evaluation of a candidate solution follows the next steps:

1. Initialize l_t and z_c to X, then, we generate 27 equidistant points in a cubic shape of length l_t centered at $(0, 0, z_c)$, in such a way that a point has the coordinate (x_i, y_i, z_c). The constraints in Eq. (5.7) are verified to be fulfilled for each point.

2. If any point does not fulfills all the constraints, Δ_s is decreased by a factor of 0.5, otherwise the length of the side of the cube l_t is updated as $l_{t+1} = l_t + \Delta_s$.

3. The cycle is repeated until Δ_s is lower than a tolerance of $1e$-8.

4. The axis z is discretized and computationally intensive search over the axis is performed in order to find the center that maximizes the regular workspace using the previous steps.

The set of optimization variables are denoted as $\boldsymbol{\alpha} = [a\ b\ d]^T$. The search range of the links lengths is $[0, 1]$ for lengths a, b and d. Joint limits and manipulability constraints are taken as in [14] and the inverse kinematics for the Delta robot is computed according to the approach in [30], which is summarized in Appendix B.1. Fig. 7.4 presents the optimal approximation of the whole workspace, as well as the enclosure regular workspace. Fig. 7.4 (a) shows the minimum and maximum workspace with the approximated optimal design. Figs. 7.4 (b), (c) and (d) show a cross section of the workspace at heights of 0.4137, 0.5940 and 0.7742 units, respectively.

TABLE 7.4: Best solution for Eq. (5.5), links lengths $\{a, b, d\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$.

EA	a	b	d	$\mathcal{F}(\boldsymbol{\alpha})$
CMA-ES	4.01e-1	5.67e-1	3.23e-2	1.80e-1
OMNI	3.64e-1	5.61e-1	2.57e-2	1.75e-1
BUMDA	3.81e-1	5.77e-1	3.67e-2	1.68e-1

Table 7.4 shows the best solution found by each algorithm for the lengths $\{a, b, c\}$ and the corresponding objective function as well as the mean and standard deviation of ten independent executions. As can be notice, the best solution is found by the CMA-ES, nevertheless, the lengths are similar for all the algorithms.

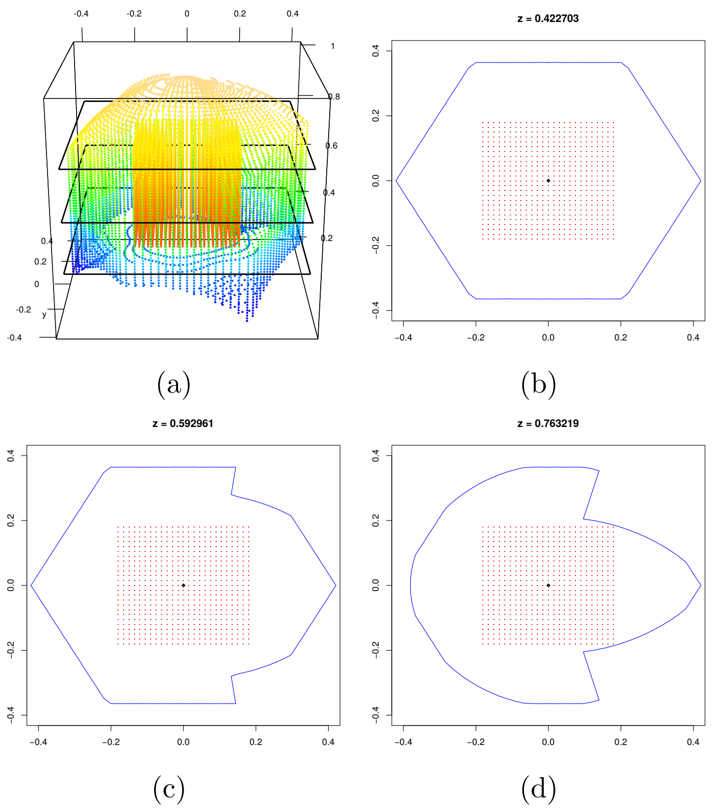


FIGURE 7.4: Workspace visualization. (a) 3D total and regular workspace. (b) Cross-section at 0.4137. (c) Cross-section at 0.5940. (d) Cross-section at 0.7742.

Figure 7.4 presents the optimal approximation of the whole workspace for the Delta robot, as well as the enclosed regular workspace. Fig. 7.4 (a) shows the minimum and maximum robot workspace with the optimal design. Figures 7.4(b), (c) and (d) show a cross section of the workspace at heights of 0.4137, 0.5940 and 0.7742 units, respectively.

7.2.2 Optimal concurrent mechanism design with a kinematic control

The second problem addressed for the Delta manipulator is defined in Section 5.3. We aim to find the geometric and control parameters to minimize the error to a tracked trajectory. In this case, we use 100 equidistant points in time, sampled from the function $\mathbf{X}_i(t) = \left\langle c_2 \sin(2.0\pi \cdot c_1), c_2 \cos(\pi \cdot c_1), c_3(c_1^4) + 0.2 \right\rangle$, where $c_1 = \frac{2.0 \cdot t - t_f}{t_f}$, $c_2 = 0.3$, $c_3 = 0.5$, $0.0 \leq t \leq 30.0\ s$ and $i = 1, \ldots, 100$. The search range of the links lengths is $[0, 1]$ for lengths a, b and d, and of the control parameters is $[0, 100]$ for gains k_p and k_d.

Table 7.5: Best solution for Eq. (5.7), links lengths $\{a, b, d\}$, control gains $\{k_p, k_d\}$ and objective value $\mathcal{F}(\boldsymbol{\alpha})$.

EA	a	b	d	k_p	k_d	$\mathcal{F}(\boldsymbol{\alpha})$
			Proportional control			
CMA-ES	4.35e-1	4.90e-1	7.38e-2	9.99e+1	-	3.71e-1
OMNI	4.40e-1	4.87e-1	7.56e-2	9.99e+1	-	3.71e-1
BUMDA	4.35e-1	4.92e-1	7.64e-2	1.00e+2	-	3.71e-1
			Proportional-derivative control			
CMA-ES	4.36e-1	4.89e-1	7.35e-2	9.99e-1	8.28e-1	8.74e-3
OMNI	4.37e-1	4.88e-1	7.66e-2	9.99e-1	8.28e-1	8.76e-3
BUMDA	4.35e-1	4.92e-1	7.36e-2	9.99e-1	8.28e-1	8.75e-3

Table 7.5 shows results obtained by the EAs. Notice that for both controls, P and PD, the best structure parameters are almost the same, meaning that those parameters might be one local optimum. Nevertheless, the proportional gain is larger for the P than for the PD controller, meaning that the P controller is using more energy in order to achieve a reasonable performance. This can be noticed in the objective function column. In addition, notice that all algorithms reach basically the same objective function value, but the CMA-ES consistently delivers a similar value for the P and PD control gains. Fig. 7.5 (a) and (c) shows the logarithmic absolute EEPE over time for the P and PD control, respectively. Figs. 7.5 (b) and (d) show in green the path undertaken by the Delta robot and the interpolated path in blue, for the P and PD controls, respectively. Fig. 7.5 (e) describes the integration of the control signal over time, in order to compare energy consumption of the P and PD controllers. As shown, the P control reaches faster the target trajectory at a higher energy consumption.

As in the previous section, in order to objectively compare the optimization methods, we use hypothesis tests to compare pairs of them via the bootstrap methodology, it is presented in Table 7.6. Each evolutionary algorithm is executed 10 times for each optimization problem under the same conditions. According to Table 7.6, the methods

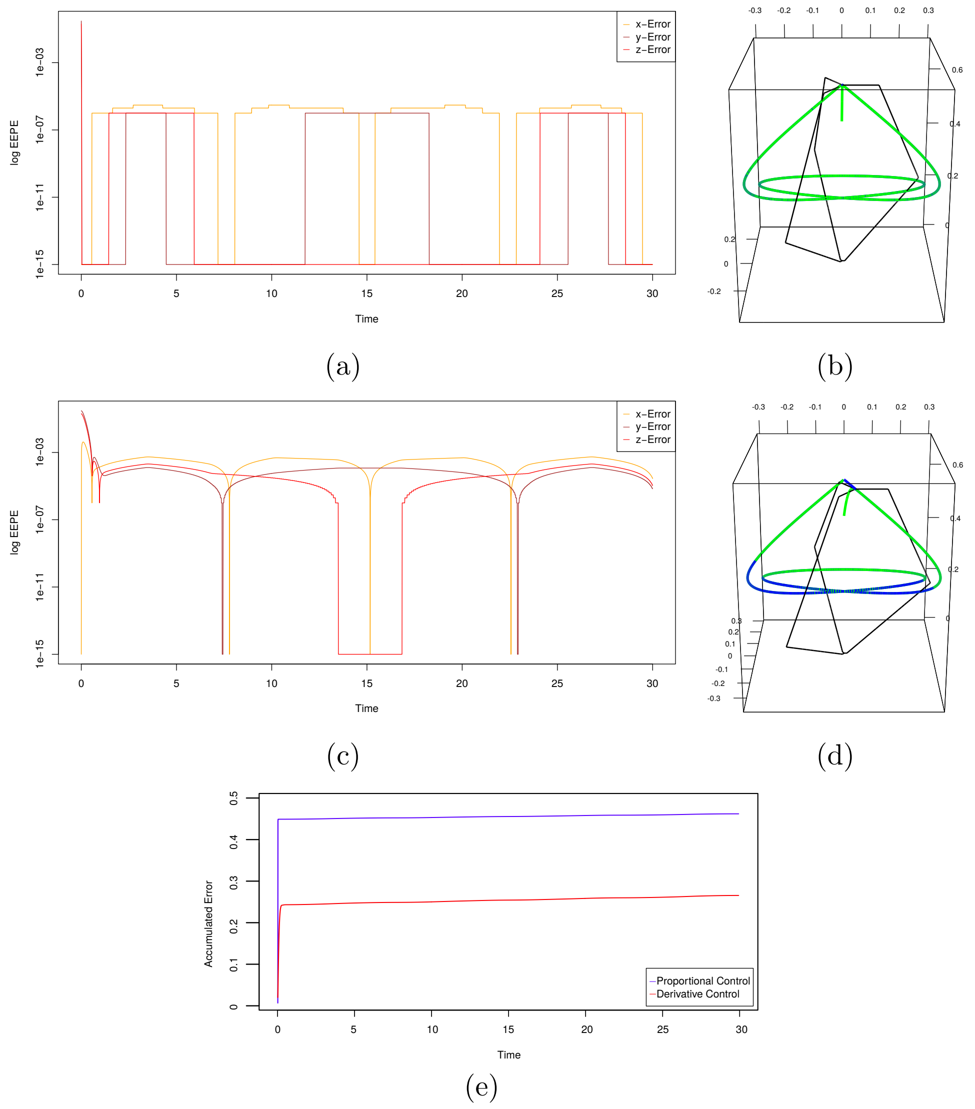


Figure 7.5: Trajectory tracking control for the Delta robot. (a) Logarithmic absolute EEPE over time for P control. (b) path tracking for P. (c) Logarithmic absolute EEPE over time for PD control. (d) path tracking for PD. (e) Absolute EEPE integral for P and PD control.

perform similar, considering the objective function values as well as their p-value. Nevertheless, we can see that CMA-ES reports the best objective function value and the smallest standard deviation for the addressed problem.

7.3 Case study: Two-link serial manipulator

Similar to the anthropomorphic manipulator, a two-link manipulator is an open-chain serial arm with movement restricted to a plane. A representation of the mechanism is shown in Fig. 7.6. Although this robot is simple, we have considered it as a case of study since it is used as a typical problem in literature for comparison purposes [23, 24, 48].

Table 7.6: Results of the hypothesis test ($\mu_A < \mu_B$) with a 5% significance for the optimization of a parallel Delta manipulator, with the objective functions described in Eq. (5.5) and Eq. (5.7).

EA$_1$ vs EA$_2$	a	b	d	k_p	k_d	$\mathcal{F}(\boldsymbol{\alpha})$
1.- Optimal maximum regular workspace mechanism design						
CMA-ES vs BUMDA	=	=	< (0.01962)	-	-	6.00e-4
CMA-ES vs OMNI	=	=	< (0.02143)	-	-	0.000
BUMDA vs OMNI	> (0.03162)	=	=	-	-	8.200e-4
2.- Optimal concurrent mechanism design with a kinematic proportional control						
CMA-ES vs BUMDA	=	=	< (0.01616)	=	-	0.000
CMA-ES vs OMNI	=	=	=	=	-	0.000
BUMDA vs OMNI	=	=	=	=	-	0.110 *
3.- Optimal concurrent mechanism design with a kinematic proportional-derivative control						
CMA-ES vs BUMDA	=	=	=	=	=	0.033
CMA-ES vs OMNI	=	=	=	> (0.000)	=	2.100e-4
BUMDA vs OMNI	=	=	=	> (0.00882)	=	0.134 *

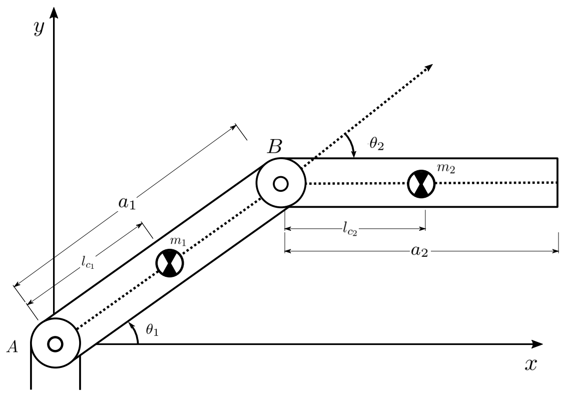


Figure 7.6: Case study: Two-link serial manipulator. Schematic of the architecture of the two-link planar manipulator.

In this section, we present the results of optimizing the objective function 5.4 using the two-link planar manipulator with simplifications of its dynamic model, as it is shown in Appendix A.2.1. In order to validate our proposed results, we present a comparison of our results using the same optimization methods like in the previous case studies and the results found in literature.

7.3.1 Minimal joint error mechanism design with a dynamic control

The first problem addressed for the two-link planar manipulator is defined in Section 5.4.1. A PID controller as shown in Appendix C.1. In the implementation, the robot dynamics is solved by a $4th$ order Runge-Kutta method.

In this case, the optimization methods approximate the optimal control parameters that minimize the error of the tracked trajectory computed as a cubic interpolation of the joint trajectory. The desired position and velocity values are $\theta_1^D(t = 2s) = 1\ rad$, $\theta_2^D(t = 2s) = 2\ rad$, $\theta_1^D(t = 4s) = 0.5\ rad$ and $\theta_2^D(t = 4s) = 4\ rad$ and $\dot{\theta}_1^D(t = 2s) = \dot{\theta}_2^D(t = 2s) = \dot{\theta}_1^D(t = 4s) = \dot{\theta}_2^D(t = 4s) = 0\ rad/s$, respectively. The search range of the control parameters is $[10, 350]$ for gains $k_p^{(j)}$, and $[0, 50]$ for gains $k_d^{(j)}$ and $k_i^{(j)}$ for $j = 1, 2$.

Table 7.7: Fixed conditions of the two-link planar manipulator.

Parameter	Link$_1$	Link$_2$
Mass (kg)	0.1	0.1
Length (m)	0.8	0.4
Inertia (kgm^2)	0.064	0.016

Table 7.7 shows the fixed parameters for the two-link planar manipulator. The dynamic model applied to this optimization problem is presented in Appendix A.2.1.

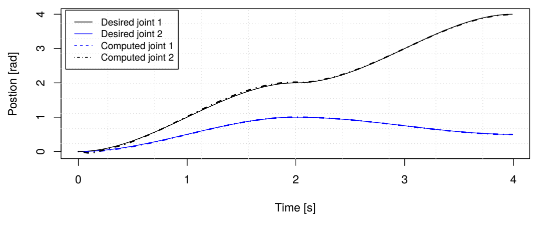
Table 7.8: Best solution for Eq. (5.10), control gains $\{k_p^{(j)}, k_d^{(j)}, k_i^{(j)}\}$ for j^{th} joint and objective value $\mathcal{F}_1$ and Eq. (5.11) $(\mathcal{F}_2)$.

EA	$k_p^{(1)}$	$k_p^{(2)}$	$k_d^{(1)}$	$k_d^{(2)}$
Known [24]	1.84e+2	1.16e+1	9.10	1.1e-1
CMA-ES	3.50e+2	1.87e+2	4.14e-14	2.07e-15
OMNI	3.49e+2	1.83e+2	4.33	9.78e-3
BUMDA	3.49e+2	1.64e+2	2.82	2.03e-1
EA	$k_i^{(1)}$	$k_i^{(2)}$	$\mathcal{F}_1^*$	$\mathcal{F}_2$
Known [24]	4.92e+1	1.62e+1	7.32	11.57
CMA-ES	5.00e+1	5.0e+1	1.46	3.80e+1
OMNI	4.97e+1	4.99e+1	1.47	3.30e+1
BUMDA	4.99e+1	4.98e+1	1.52	3.33e+1

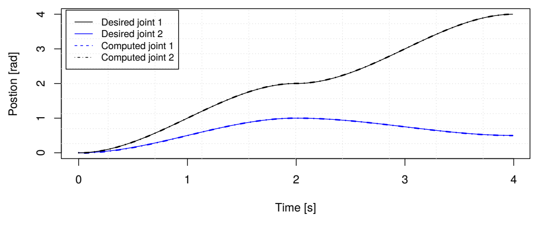
A comparison of the results delivered by our approach versus those in literature [24] are shown in Table 7.8. In their work, a multiobjective optimization method is used in order to obtain a set of non-dominated candidate solutions. Therefore, the non-dominated candidate with a better performance of $\mathcal{F}_1^*$ is selected as the presented solutions. In the comparison, we present the best configuration of control gains for the non-dominated solution reported in literature and ours, as well as the objective function $\mathcal{F}_1^*$ which is the total error of the tracked trajectory, described by Eq. (5.10) and the objective function $(\mathcal{F}_2)$ which computes the total energy applied to the mechanism, described by Eq. (5.11). As can be seen, the comparison shows that the EAs are able to set similar control gain configurations that minimize $\mathcal{F}_1^*$, pointing out that they are close to the optimal solution. Compared to the non-dominated solution presented in literature [24],

Best known solution

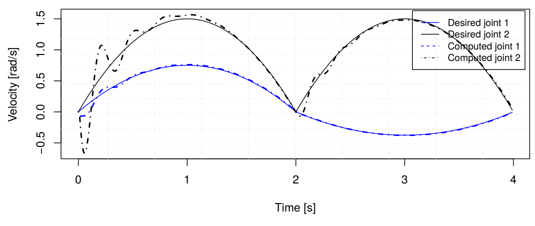
Best found solution



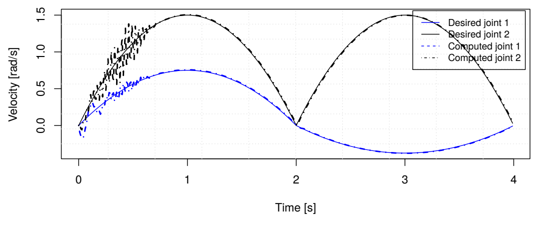
(a)



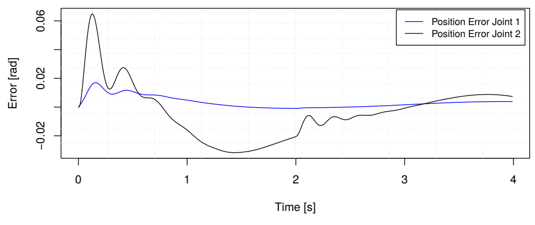
(b)



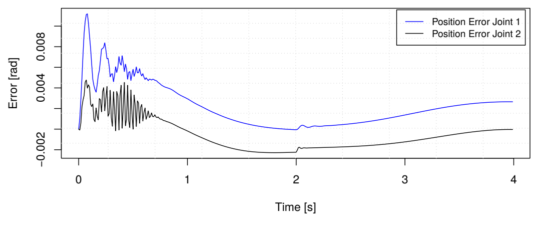
(c)



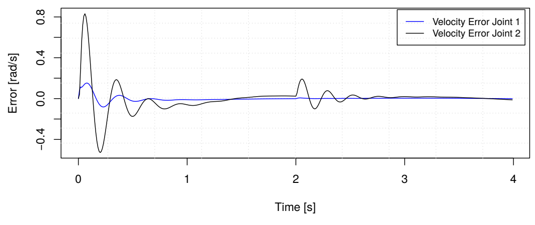
(d)



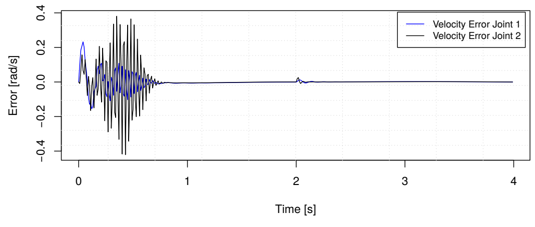
(e)



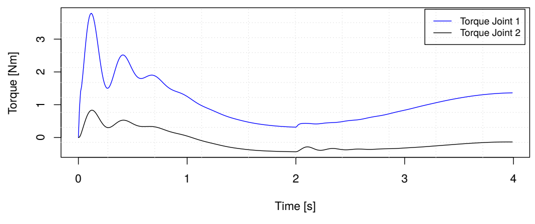
(f)



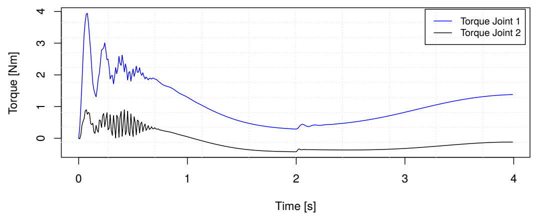
(g)



(h)



(i)



(j)

FIGURE 7.7: Comparison of the best found and known control design for minimal error optimization. (a) Known optimal joints position. (b) Found optimal joints position. (c) Known optimal joints velocity. (d) Found optimal joints velocity. (e) Known optimal joints position error. (f) Found optimal joints position error. (g) Known optimal joints velocity error. (h) Found optimal joints velocity error. (i) Known optimal joints torque. (j) Found optimal joints torque.

the EAs found solutions with a better performance, with the disadvantage of increasing the second objective function $\mathcal{F}_2$, thus, the tracking error is reduced by increasing the energy consumption. We must assume that our solution is a dominated solution.

A graphical comparison of the known optimal solution in [24] is shown in Fig. 7.7, where (a) and (b) show the position error for each joint throughout time, (c) and (d) show the velocity error for each joint throughout time and finally, (e) and (f) show the applied torque of each joint throughout time, for the known solution and the best solution found, respectively.

Notice Fig. 7.7 (e) and Fig. 7.7(f), the found solution tracks the path faster and with lesser error than the known solution, at the cost of higher energy consumption, as it is shown in Fig. 7.7(i) and Fig. 7.7(j). The oscillations in the first time steps are due to the compensations that the control induces via velocity errors as it is shown in Fig. 7.7(h), because of the velocity gradient.

7.3.2 Minimal energy mechanism design with a dynamic control

The second design problem addressed for the two-link planar manipulator is defined in Section 5.4.2. The evolutionary algorithms deal with the minimization of the total energy applied to the mechanism in order to find the approximated optimal control gains that tracks a desired trajectory, which is built by a cubic interpolation. The desired position and velocity values are given by $\theta_1^D(t = 2s) = 1\ rad$, $\theta_2^D(t = 2s) = 2\ rad$, $\theta_1^D(t = 4s) = 0.5\ rad$ and $\theta_2^D(t = 4s) = 4\ rad$ and $\dot{\theta}_1^D(t = 2s) = \dot{\theta}_2^D(t = 2s) = \dot{\theta}_1^D(t = 4s) = \dot{\theta}_2^D(t = 4s) = 0\ rad/s$, respectively. The search range of the control parameters is $[10, 350]$ for gains $k_p^{(j)}$, and $[0, 50]$ for gains $k_d^{(j)}$ and $k_i^{(j)}$ for $j = 1, 2$. The fixed parameters for the mechanism are presented in Table 7.7.

TABLE 7.9: Best solution for Eq. (5.11), control gains $\{k_p^{(j)}, k_d^{(j)}, k_i^{(j)}\}$ for j^{th} joint and objective value $\mathcal{F}_2$ and Eq. (5.10) ($\mathcal{F}_1$).

EA	$k_p^{(1)}$	$k_p^{(2)}$	$k_d^{(1)}$	$k_d^{(2)}$
Known [24]	1.84e+2	1.12e+1	8.85	2.7e-1
CMA-ES	3.49e+2	1.00e+1	1.48e+1	1.09
OMNI	3.49e+2	1.00e+1	1.54e+1	1.07
BUMDA	3.07e+2	1.01e+1	1.39e+1	1.24
EA	$k_i^{(1)}$	$k_i^{(2)}$	$\mathcal{F}_1$	$\mathcal{F}_2^*$
Known [24]	4.94e+1	1.62e+1	7.45	9.93
CMA-ES	5.00e+1	7.43	9.65	9.21
OMNI	3.58e+1	6.94	9.87	9.20
BUMDA	2.49e+1	8.44	9.58	9.32

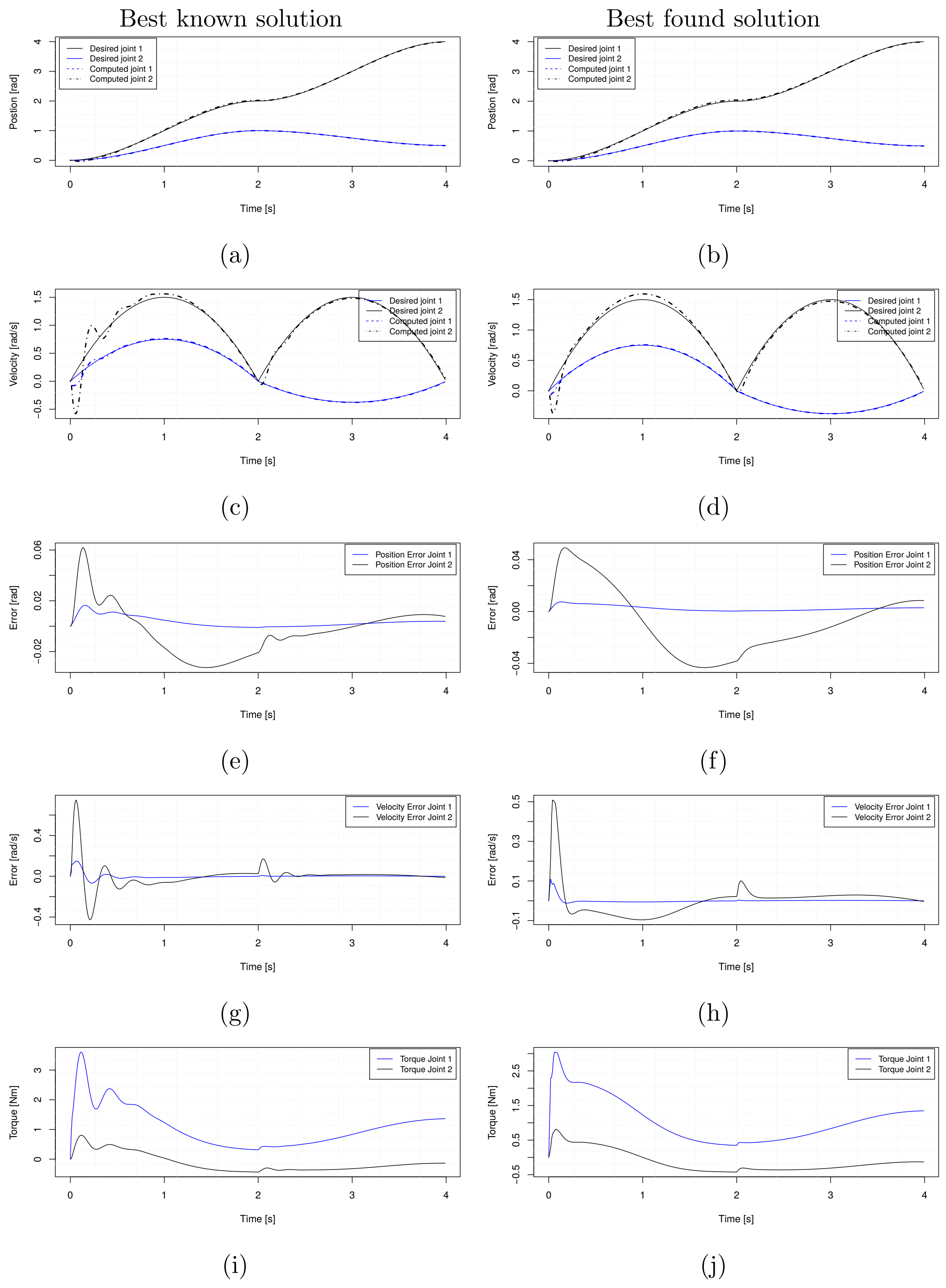


FIGURE 7.8: Comparison of the best found and known control design for minimal energy optimization. (a) Known optimal joints position. (b) Found optimal joints position. (c) Known optimal joints velocity. (d) Found optimal joints velocity. (e) Known optimal joints position error. (f) Found optimal joints position error. (g) Known optimal joints velocity error. (h) Found optimal joints velocity error. (i) Known optimal joints torque. (j) Found optimal joints torque.

A comparison of the results from the EAs and the results in literature [24] are shown in Table 7.9. As in the previous optimization problem, a multiobjective optimization method is used in order to obtain a set of non-dominated candidate solutions and the non-dominated solution with the better performance of $\mathcal{F}_2^*$ is selected as the presented solution. The comparison shows that some control gains found by the EAs are similar, in order to minimize the objective function $\mathcal{F}_2^*$, pointing out that there are local minima near the optimal solution. Comparing the non-dominated solution presented in literature [24], the found solutions from the EAs have better performance, with the disadvantage of increasing the second objective function $\mathcal{F}_1$, thus, the energy consumption is reduced while the tracking error is increased. We must assume that our solutions are dominated.

A graphic comparison of the known optimal solution for the design problem and the best found optimal solution is shown in Fig. 7.8, where (a) and (b) shows the tracked and generated path, (c) and (d) shows the velocity of the tracked and generated path, (e) and (f) show the position error for each joint throughout time, (g) and (h) show the velocity error for each joint throughout time and, (i) and (j) show the applied torque of each joint throughout time, for the known solution and the best optimal solution, respectively.

Notice that as mentiones above, the found solution allows the mechanism to track the trajectory with less energy consumption as can be appreciated in Figs. 7.7 (i) and (j), with the cost of an increment of the tracking errors, as shown in Figs. 7.7(e)-(h).

7.3.3 Optimal simultaneous mechanism design with a dynamic control

In this case study, the simultaneous optimization problem, presented in Section 5.4, is used to optimize the parameters of a PID control. The main purpose of employing these fitness functions is to minimize the tracking errors between actual and desired trajectories and also, to minimize the energy consumption.

The polynomial trajectory is given by $\theta_i^D(t) = \theta_i^D(t_0) + (6r_{time}^5 - 15r_{time}^4 + 10r_{time}^3)(\theta_i^D(t_f) - \theta_i^D(t_0))$, where $r_{time} = t_0/t_f$, $\theta_1^D(t_0) = \theta_2^D(t_0) = 0$ for $t_0 = 0$ when and $\theta_1^D(t_f) = \pi/2$ and $\theta_2^D(t_f) = \pi/6$ for $t_f = 2$. The search range of the control parameters is $[0, 400]$ for gains $k_p^{(j)}$, $k_d^{(j)}$ and $k_i^{(j)}$, for $j = 1, 2$. The fixed parameters of the manipulator are presented in Table 7.7.

A comparison of the results by the EAs and the found results in literature [48] are shown in Table 7.10. The comparison shows that CMA-ES and Omni-optimizer are able to set similar control gain configurations that minimize $\mathcal{F}^*$, pointing out that they are close to the optimal solution. As for the BUMDA, some control gains are different from one

TABLE 7.10: Best solution for Eq. (5.9), using weights $\omega_1 = \omega_2 = 0.5$, control gains $\{k_p^{(j)}, k_d^{(j)}, k_i^{(j)}\}$ for j^{th} joint and objective value $\mathcal{F}^*$, Eq. (5.10)($\mathcal{F}_1$) and Eq. (5.11)($\mathcal{F}_2$).

EA	$k_p^{(1)}$	$k_p^{(2)}$	$k_d^{(1)}$	$k_d^{(2)}$	$k_i^{(1)}$
Known [48]	$3.63e$+2	$3.58e$+2	6.79	$1.77e$+1	$3.54e$+1
CMA-ES	$4.00e$+2	$4.00e$+2	$4.48e$+1	$1.15e$+1	$4.00e$+2
OMNI	$3.99e$+2	$3.99e$+2	$4.47e$+1	$1.17e$+1	$3.99e$+2
BUMDA	$3.99e$+2	$3.99e$+2	$2.13e$+2	8.27	$3.84e$+2

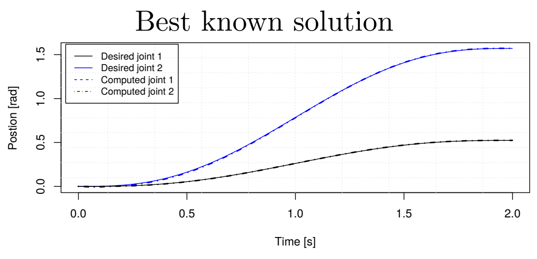
EA	$k_i^{(2)}$	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}^*$
Known [48]	$3.07e$+2	$4.36e$-3	$6.18e$-3	$5.27e$-3
CMA-ES	$4.00e$+2	$3.56e$-3	$4.55e$-3	$4.06e$-3
OMNI	$3.99e$+2	$3.56e$-3	$4.55e$-3	$4.06e$-3
BUMDA	$3.99e$+2	$3.11e$-3	$5.04e$-3	$4.07e$-3

found with the other two EAs, thus, we can assume that a local minimum is near the optimal solution. Compared to the solution presented in literature [48], the EAs found solutions with a better performance for both objectives functions $\mathcal{F}_1$ and $\mathcal{F}_2$, therefore, the presented solution are able to minimize the tracking error as well as the energy consumption.

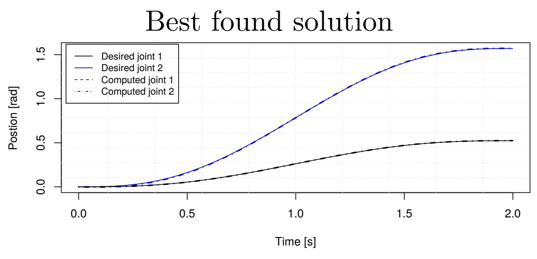
A graphic comparison of the literature optimal solution for the design problem and the best found optimal solution is shown in Fig. 7.9, where (a) and (b) shows the tracked and generated path, (c) and (d) shows the velocity of the tracked and generated path, (e) and (f) show the position error for each joint throughout time, (g) and (h) show the velocity error for each joint throughout time and, (i) and (j) show the applied torque of each joint throughout time, for the known solution and the best optimal solution, respectively.

7.3.4 Optimal concurrent mechanism design with a dynamic control

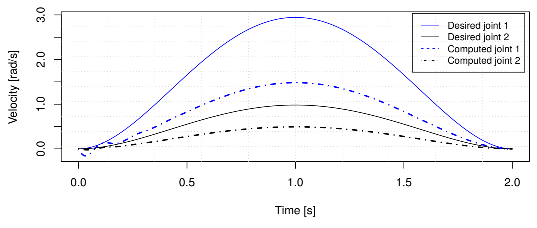
In this case we present a simultaneous optimization of the control parameters and masses of counterweights at the rear end of the links of a two-link planar manipulator, like in [23]. In this case, an adaptive PD controller, as described in Appendix C.2, is applied to the mechanism to track a cubic polynomial interpolation of the angles, with the following given points where $\theta_1^D(t_0) = \theta_1^D(t_0) = 0$, $\theta_1^D(t_f) = \pi/2$, $\theta_1^D(t_f) = \pi/6$ for the desired position, and $\dot{\theta}_1^D(t_0) = \dot{\theta}_2^D(t_0) = \dot{\theta}_1^D(t_f) = \dot{\theta}_2^D(t_f) = 0\ rad/s$ for the desired velocities. The search range of the control parameters is $[0, 100]$ for gains $k_p^{(j)}$ and $k_d^{(j)}$, $[0, 1]$ for $a^{(j)}$ and b^j, and $[0, 20]$ for the counterweigth masses $m_\omega^{(j)}$, where $j = 1, 2$.



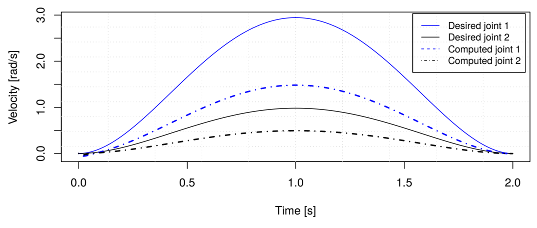
(a)



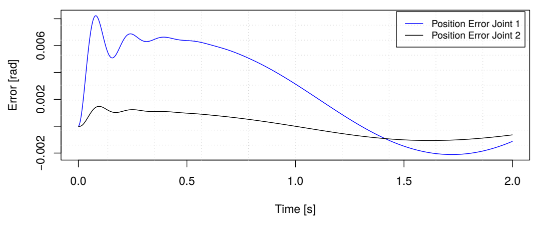
(b)



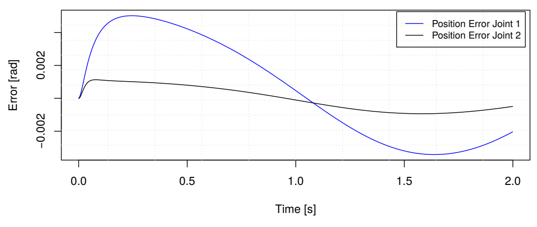
(c)



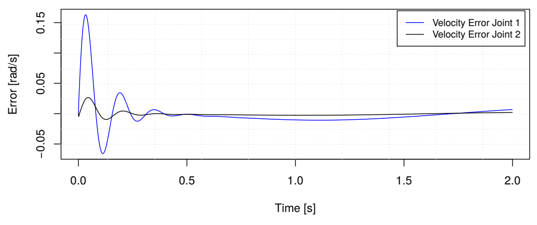
(d)



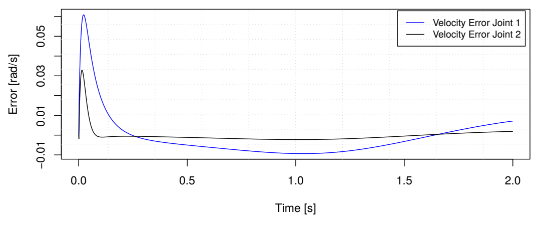
(e)



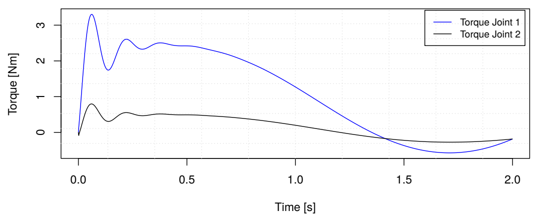
(f)



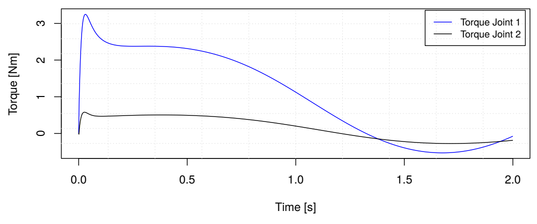
(g)



(h)



(i)



(j)

FIGURE 7.9: Comparison of the best found and known control design for simultaneous optimization. (a) Known optimal joints position. (b) Found optimal joints position. (c) Known optimal joints velocity. (d) Found optimal joints velocity. (e) Known optimal joints position error. (f) Found optimal joints position error. (g) Known optimal joints velocity error. (h) Found optimal joints velocity error. (i) Known optimal joints torque. (j) Found optimal joints torque.

TABLE 7.11: Fixed conditions of the two-link planar manipulator.

Parameter	Link$_1$	Link$_2$
Mass (kg)	0.5	0.5
Length (m)	1.0	1.0
Inertia (kgm^2)	0.0833	0.0833
Counterweight radius (m)	0.4	0.4

Table 7.11 shows the fixed parameters for the two-link planar manipulator with counterweight. The dynamic model applied to this optimization problem is presented in Appendix A.2.2.

TABLE 7.12: Best solution for Eq. (5.9), using weights $\omega_1 = 0.8$ and $\omega_2 = 0.2$, control gains $\{k_p^{(j)}, k_d^{(j)}, a^{(j)}, b^j\}$ and counterweigth masses $m_\omega^{(j)}$ for j^{th} joint and objective value $\mathcal{F}^*$.

EA	$k_p^{(1)}$	$k_p^{(2)}$	$k_d^{(1)}$	$k_d^{(2)}$	$a^{(1)}$	$a^{(2)}$
Known [23]	3.45e+1	9.61e+1	8.17	1.84e+1	6.43e-2	5.81e-2
CMA-ES	0.00	0.00	3.91e+1	6.21e-2	9.73e-1	1.21e-1
OMNI	2.27e+1	2.86e-3	7.82e+1	4.96e-3	1.78e-1	4.20e-1
BUMDA	5.40e+1	9.40e+1	5.32e+1	4.23e+1	5.81e-1	1.89e-2
EA	$b^{(1)}$	$b^{(2)}$	$m_\omega^{(1)}$	$m_\omega^{(2)}$	$\mathcal{F}^*$	
Known [23]	9.47e-2	4.78e-2	1.25e+1	2.5	1.97e-1	
CMA-ES	0.00	1.00	9.48	1.97	2.30e-2	
OMNI	5.95e-1	4.01e-1	9.62	2.00	2.87e-2	
BUMDA	4.61e-1	6.31e-1	9.34	1.85	2.67e-2	

A comparison of the found results by the EAs and the results in literature [23] are shown in Table 7.10. The comparison shows that EAs are not able to set similar control gains configurations, even when the presented solutions delivers a better performance than the solutions found in literature [23]. This behavior can be attributed to the multimodality of the problem, thus, the EAs stagnate in a local minima and it is not possible to ensure that they are able to approach the optimal solution.

The graphic comparison of the reported solution in literature for the design problem and the best found optimal solution is shown in Fig. 7.10, where (a) and (b) shows the tracked and generated path, (c) and (d) shows the velocity of the tracked and generated path, (e) and (f) show the position error for each joint throughout time, (g) and (h) show the velocity error for each joint throughout time and, (i) and (j) show the applied torque of each joint throughout time, for the known solution and the best optimal solution, respectively. In Figs. 7.10 (b) and (d), the presented solution is able to track more accurately than the solution found in literature, therefore, a lesser error as seen

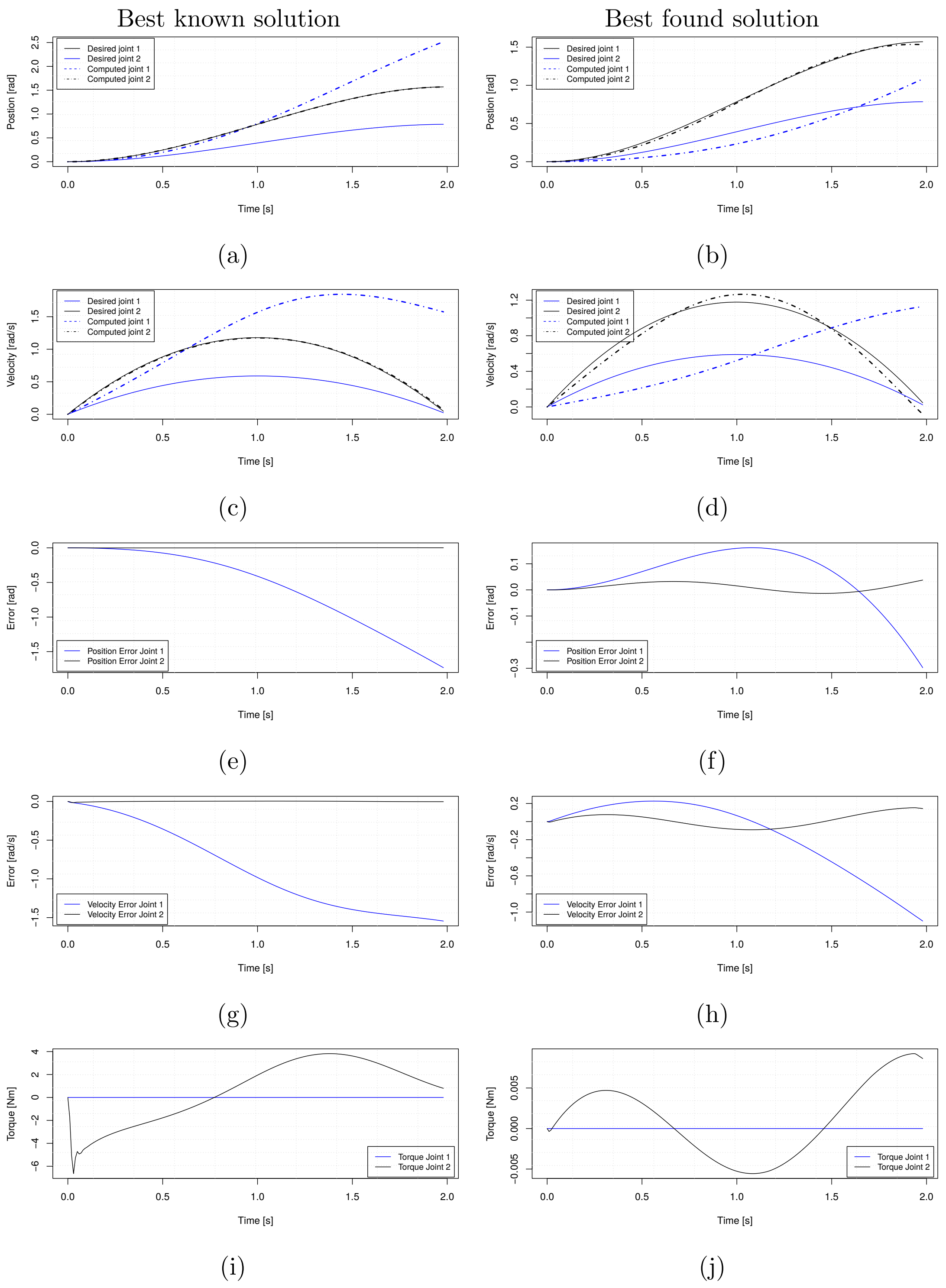


Figure 7.10: Comparison of the best found and known design mechanism for concurrent optimization. (a) Known optimal joints position. (b) Found optimal joints position. (c) Known optimal joints velocity. (d) Found optimal joints velocity. (e) Known optimal joints position error. (f) Found optimal joints position error. (g) Known optimal joints velocity error. (h) Found optimal joints velocity error. (i) Known optimal joints torque. (j) Found optimal joints torque.

in Figs. 7.10 (e)-(h). The counterweights masses allow to the manipulator to track the desired trajectory with lower energy consumption as presented in Figs. 7.10 (i) and (j).

TABLE 7.13: Hypothesis testing ($\mu_A < \mu_B$) with a 5% significance for the optimization of a two-link planar manipulator for different objective function described in Eq. (5.10), Eq.(5.11) and Eq.(5.9).

EA$_1$ vs EA$_2$	$k_p^{(1)}$	$k_p^{(2)}$	$k_d^{(1)}$	$k_d^{(2)}$	$k_i^{(1)}$	$k_i^{(2)}$	$m_\omega^{(1)}$	$m_\omega^{(2)}$	$\mathcal{F}$
1.- Minimal joint error mechanism design with a dynamic control									
CMA-ES vs BUMDA	> (0.000)	> (0.000)	< (0.000)	< (0.000)	> (0.000)	> (0.000)	-	-	0.000
CMA-ES vs OMNI	> (0.000)	> (0.000)	< (0.000)	< (0.000)	> (0.000)	> (0.000)	-	-	0.000
BUMDA vs OMNI	=	< (0.000)	> (0.00021)	=	=	< (0.00212)	-	-	0.128 *
2.- Minimal energy mechanism design with a dynamic control									
CMA-ES vs **BUMDA**	=	> (0.000)	=	< (0.000)	< (0.00052)	< (0.000)	-	-	0.000
CMA-ES vs OMNI	=	> (0.01355)	=	< (0.000)	< (0.00293)	< (0.000)	-	-	0.015
BUMDA vs OMNI	=	< (0.01409)	=	< (0.01480)	=	=	-	-	0.014
3.- Optimal simultaneous mechanism design with a dynamic control									
CMA-ES vs BUMDA	> (0.000)	> (0.000)	< (0.000)	> (0.00021)	> (0.000)	> (0.000)	-	-	0.000
CMA-ES vs OMNI	> (0.000)	> (0.000)	> (0.00197)	< (0.00194)	> (0.000)	> (0.000)	-	-	0.000
BUMDA vs OMNI	=	> (0.00037)	> (0.000)	< (0.00001)	=	> (0.000)	-	-	0.000
4.- Optimal concurrent mechanism design with a dynamic control									
CMA-ES vs BUMDA	> (0.02483)	> (0.000)	=	=	-	-	< (0.000)	< (0.000)	0.000
CMA-ES vs OMNI	=	> (0.000)	=	< (0.000)	-	-	=	< (0.02321)	0.023
BUMDA vs **OMNI**	=	=	=	< (0.000)	-	-	> (0.000)	> (0.000)	0.000

Finally, Table 7.13 shows the comparison of the optimization algorithms using the two-link planar manipulator with counterweigths, in which, each evolutionary algorithm is executed 50 times for each optimization problem, under the same conditions, and an intensive sampling technique (bootstrap) is applied for hypothesis testing. As can be seen, the CMA-ES has a better performance in 75% of the presented experiments, but in comparison to the static and kinematical problems with more complex mechanisms, presented in Tables 7.3 and 7.6, the optimized parameters are not similar, implying that the dynamic optimization problems are more complex, due to the possible multimodality on the objective functions.

7.4 BFGS vs EAs

For the sake of comparison, a gradient-based optimization method is used to solve the optimization problems, presented in section 7.3.1, 7.3.2 and 7.3.4. In this case, the objective functions are coded in MATLAB® , where the function *"minunc"* is used as the optimization tool. This function is a nonlinear programming solver, that finds the minimum of a convex problem. In this case we use the, a quasi-Newton algorithm is set as the optimization method, which uses the Broyden-Fletcher-Goldfarb-Shanno (BFGS) Quasi-Newton method with a cubic interpolation for the line search. In particular, this quasi-Newton method differs from others, mainly, in the formula used for updating the inverse of the Hessian matrix approximation. For these experiments, the initial solutions are generated randomly with a uniform distribution.

TABLE 7.14: Best solution Eq. (5.10) ($\mathcal{F}_1$), Eq. (5.11) ($\mathcal{F}_2$) and for Eq. (5.9) ($\mathcal{F}_3$), using weights $\omega_1 = 0.5$ and $\omega_2 = 0.5$.

EA	BFGS	CMA-ES $\overline{\mathcal{F}} + \sigma$	OMNI $\overline{\mathcal{F}} + \sigma$	BUMDA $\overline{\mathcal{F}} + \sigma$
$\mathcal{F}_1$	2.22	$1.47 + 0.00$	$4.01e{+}2$ $+1.97e{+}3$	$2.66 + 4.55$
$\mathcal{F}_2$	$2.70e{+}2$	$4.07 + 1.99$	$8.08e{+}2$ $+1.97e{+}3$	$9.77 + 1.98e{-}1$
$\mathcal{F}_3$	1.21	$4.06e{-}3$ $+4.15e{-}6$	$4.10e{-}3$ $+7.04e{-}5$	$4.07e{-}3$ $+3.40e{-}6$

It is well known that gradient-based optimization methods are widely biased by the initial solution and the multimodality of the optimization problem [49]. Thus, gradient-based methods could not find a globally optimal solution and because of it could be trapped in a local minimum, depending on the initial solution. In addition, the BFGS uses a numerical approximation for the Hessian and Jacobian [50]. This causes that the function evaluations of the design problem increases drastically with respect to computing the analytical Hessian and Jacobian.

In Table 7.14, the approximation of the optimal fitness function value for BFGS and the mean and standard deviation for the EAs are shown. It can be seen that the BFGS does not reach the solutions presented in Tables 7.8, 7.9 and 7.10, obtained by the EAs. In most of the cases, the EAs reach a better solution approximation than BFGS.

7.5 Proposed parallelization results

The proposed parallelization for an EDA, presented in Section 6.1, is tested using the optimization problems described in Section 7.2.1 and 7.2.2, where the design and control parameters of a Delta parallel manipulator are optimized to maximize a regular workspace and minimize the error of a tracked trajectory using a kinematic control, respectively.

For this evaluation, Open MPI is used as the Message Passing Interface for distributed memory parallelization in the parallelized EDA. For each case of the design problem, the optimization algorithm is executed 5 times, with different numbers of raised process. The algorithms stop when: a maximum of $1e{+}5{\cdot}p$ evaluations, for p optimization variables. To verify the feasibility of the proposed parallelization, the population is set to $1e{+}3{\cdot}p$. The EDA used is the BUMDA and the average-based weighting from Section 6.2.1 is used to approach the global distribution.

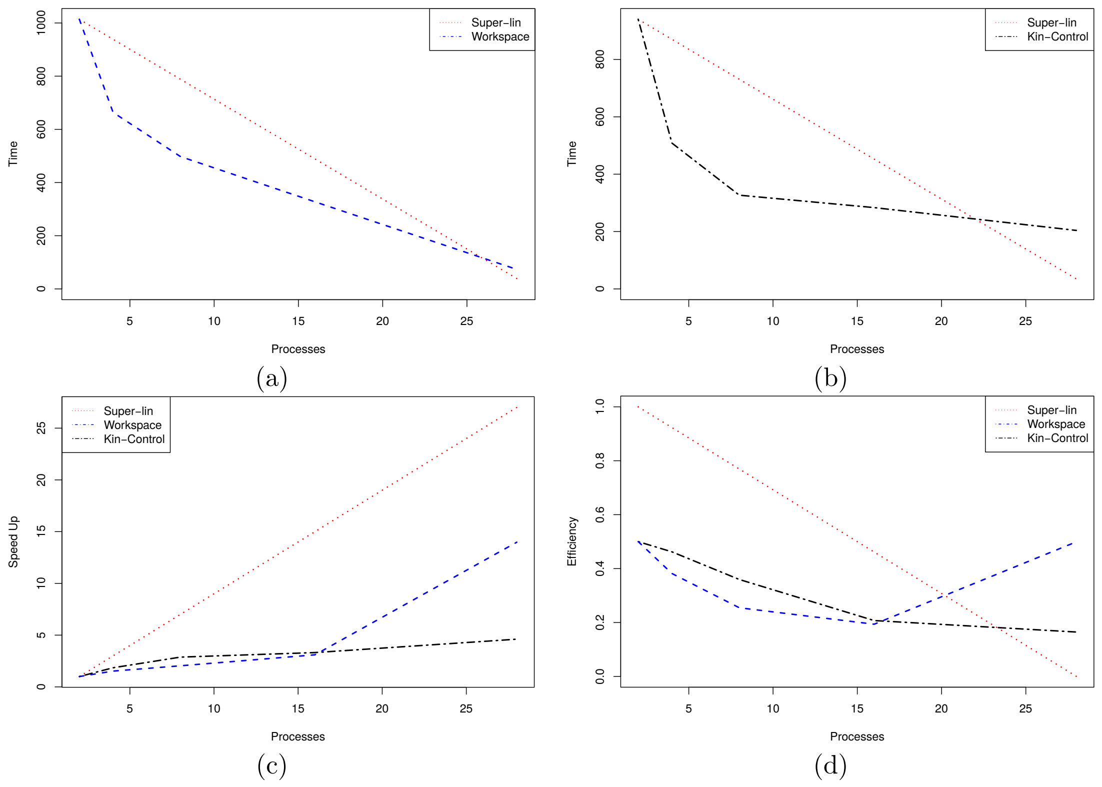


Figure 7.11: (a) Runtimes for the maximum regular workspace problem. (b) Runtimes for the minimal trajectory tracking error for a kinematic problem. (c) Speed up. (d) Efficiency.

The parallelization results are presented in Fig. 7.11, where, Fig. 7.11 (a) and (b) show the execution times for the parallelization of the regular workspace and kinematic control optimization problems, respectively. In Fig. 7.11 (c) and (d), the speed up and the efficiencies of parallelization for both objective functions are presented, respectively. As mentioned by [51], the linearization of the speed up of the parallelized EAs is theoretically possible in a heterogeneous computing system and even desired. This is a remarkable opinion, meaning that the algorithms that behave in such way are highly efficient.

Chapter 8

Conclusions and future work

In this chapter, the conclusions of the case studies, optimization problems and results presented in the previous chapter are presented. We divide the conclusion in four main sections, regarding to the applied optimization methods, the study cases, the optimization problems and the optimization results, where the comparison Tables 7.13, 7.3 and 7.6 and the results from each optimization problem in Sections 7.3, 7.1 and 7.2 are of great uses to elucidate our conclusions. Additionally, possible and interesting lines of research and future work are presented, which can benefit from the application presented in Section 6.3 *CODeMe*.

8.1 Conclusions

In this work, we have reviewed four mathematical models for optimal mechanism design. We have shown that the problems can be addressed under a unified framework, although they are different in the goal, optimization variables and complexity. This means that the problem is to find a set of parameters $\boldsymbol{\alpha} \in \mathbb{R}^p$, which are evaluated via a computational simulation of the mechanism.

8.1.1 Conclusions: Optimization methods

In order to use the same methodology for different problems and mechanisms, the optimization algorithm must be independent of the type of mechanism and numerical simulation performed. Because of the kinematic, structural and topological dependencies of the optimization problems, the proposed solution method should not be an optimization method for convex problems. Hence, we propose to use a set of evolutionary algorithms, which are from the family of black-box optimization. The optimization methods are compared using an intensive sampling method. This allows us to determine which optimization method is most suitable for each optimization problem.

Let us recall the results of comparison between algorithms for the ten optimization problems presented. In most of the cases, the CMA-ES consistently delivers the best solution and 90% of the cases, it statistically outperforms the other algorithms. Thus, we suggest using CMA-ES as a basic optimization method to solve the optimization problems of kinematically complex mechanisms. There are limitations on the CMA-ES, where the calculation of the parameters estimator is computationally expensive. Meanwhile, the calculation of the parameters estimator of BUMDA uses less memory and they are not as computationally expensive as the CMA-ES. Therefore, the computing can be done in situ, and even incrementally, where a new candidate solution is generated at each iteration and a set of thresholded solutions are saved in order to upgrade the estimators of the mean and variance of the Gaussian distribution. This reduces storage and computational resources without deteriorating the solution. Additionally, if the optimization problem takes into account a general method which uses continuous and discrete variables, constraints and mono or multi-objective, we suggest using the Omni-optimizer.

8.1.2 Conclusions: Case studies

In this work, we use as case studies a set of serial and parallel manipulators. The kinematic, differential kinematic and dynamic models of the two-link planar manipulator, the anthropomorphic serial manipulator and the Delta parallel manipulator are implemented. The implemented application stands the flexibility to change the models from the presented mechanisms, allowing to easily introduce and attach more models or commercial software, being the simulation one of the most difficult parts of the optimization problem.

We did a review of related work presented in literature, to the best of our knowledge, there is not a categoraization of optimization problems according to the model, neither a comparison of the most applied optimization methods nor a stablished benchmark of mechanisms and optimizations problems in order to prove the competitivity of concurrent synthesis methods and designed methodologies, comparing them to others in the state-of-the-art. We took a first step toward by categorizing and including optimization problems and mechanisms commonly used, and comparing the proposed optimization methods with those presented in literature.

8.1.3 Conclusions: Optimization problems

Based on the best algorithm, the results in the tables of Chapter 7 can also be used to infer which optimization parameters does not reach the best optimum approximation, due to a different distribution pattern. This allows us to know over which parameters an intensive search should be performed or how difficult the objective function is. As we may see, the most difficult problems to solve are the dynamic control problems. For this cases, if we compare the known solutions from the literature with our found solutions, we offer better results with the natural compromise of increasing either the tracking error or the energy consumption corresponding to the optimization problem, except for the problem of simultaneous and concurrent optimization, where we obtain better results than those in literature for both optimization problems.

8.1.4 Conclusions: Optimization results

In this work, we present a general way to deal with the problems. As can be observed in the presented results for the serial anthropomorphic, the Delta parallel manipulator and the two-link planar manipulator designed by the algorithms, successfully performs the required task. EAs can not guarantee to converge to the optimum, but it is worth to notice that the different algorithms deliver similar solutions in the decision variables. In consequence, we can argue that it is possible that the solutions are, at least, one of the best local optima.

Is worth to mention that the introduced methodology, presented in section 6, and results from Section 7.2, where the concurrent optimization of a static and kinematic optimization problems for the Delta parallel robot are shown, were published in the IEEE Latin American Conference on Computational Intelligence (LA-CCI) 2016 [52].

8.2 Future Work

In the context of the variety of mechanism and optimization methods, a first possibility is to extend the number of optimization problems and mechanisms as well as optimization methods to the implementation of the software. In relation to the objectives of the optimization problems, application of multiobjective optimization methods is the second possibility. Additionally, we would address research lines of interest:

1. **Walking robots optimization.** The optimization methods in the software could be used for training a mechanism (not-anchored two-link robot, 2D mobile robot,

biped robot) to walk and follow a trajectory in the least amount of time or energy consumption. Or even, to find the feet trajectories of minimum energy consumption. This topic has been studied in literature, but the application of the methods already implemented could simplify the implementation of an application to solve this problem, through evolutionary optimization methods.

2. **Path/trajectory planning.** The planning of a path/trajectory that must be tracked by a mechanism is a classic problem in robotics, but this line of research is not just about getting control parameters that best follow the route, the route must be selected in order to minimize a given criterion and also minimize energy consumption and fulfill several constraints, and travel time.

3. **Topological optimization.** The structural/topological optimization is a topic that has been studied in recent years, in which the main objective is to generate structures subject to applied forces and to satisfy volume restrictions. A similar idea can be used in robotics: the optimization of each link to determine their inertial properties can be done by the addition or subtraction of material subject to minimize energy consumption, for instance. This study can also be enriched using a multiobjective optimization that minimizes the energy and tracking error, through inertial optimization and fulfilling the topological optimization conditions, described previously.

4. **Prosthetic optimization.** One of the main motivations of this work is the use of robotic components in medicine, where actuated mechanisms have been increasing its use. Taking into account the research lines described above, it is possible to optimize robotic components, which help users with physiological disabilities. This project could look for optimizing the robotic prostheses through an inertial and topological optimizations in order to provide the mechanism that better fits the user.

Appendix A

Serial manipulators

The solution of the optimization problem described in Section 5, relies on an explicit mathematical model of a robot manipulator, where its structure and/or control parameters are subject to optimization. Some mathematical tools are needed in order to derive the required models.

This chapter is dedicated to present two different goals to model a serial manipulator. The first goal is the derivation of a general approach based on linear algebra of the direct kinematics modeling. This allows the end-effector position and orientation to be expressed as a function of the joint variables of the mechanical structure with respect to a reference frame. The second goal of this chapter is to present a method to derivate the equations of motion of a manipulator in the joint space (dynamic modeling). The method is based on the Lagrange formulation. For more details on this topic, the reader is referred to classical literature as [26–28].

A.1 Serial manipulator kinematic

Firstly, let us present a general definition of the serial manipulators, which are mechanisms that can be represented as an open kinematic chain, that are composed of n links and $n + 1$ joint links, since each joint connects two links. The links are numbered from 1 to n, and the joints are numbered from 0 to n, starting from the base.

A.1.1 Forward Kinematics

The forward kinematics represents the mapping from a joint space to the Cartesian space, which can be expressed as:

$$\mathbf{r} = f(\boldsymbol{\theta}) \tag{A.1}$$

where $\mathbf{r} \in \mathbb{R}^d$ represents the position and orientation of the end-effector and $\boldsymbol{\theta} \in \mathbb{R}^n$ represents the joint variables. Usually, $d = 6$ (3 translation and 3 rotation parameters) and n is the number of joints.

A commonly used convention for selecting frames of reference in robotics is the Denavit-Hartenberg representation. The Denavit-Hartenberg parameters (also called DH parameters) are the minimum number of parameters that allows the representation of the spatial relationship between reference frames attached to the link of a kinematic chain.

The coordinate frames are attached to the joints between two links such that one transformation is associated with the motion along the axis of rotation z_i of each joint, and the second transformation is associated with the motion on the axis x_i define along each link. The ordered multiplication of coordinate transformations along a serial robot consisting of n links from the kinematics equations of the robot:

$$\mathbf{T}^i_j = \mathbf{A}_i \mathbf{A}_{i+1} \dots \mathbf{A}_j \tag{A.2}$$

where $\mathbf{T}^i_j$ is the transformation that expresses the poses of link j from the reference frame of joint i and $\mathbf{A}_i$ is the homogeneous transformation that represents the relative pose between the reference frames attached to joint i and its previous joints. Each relative transformation can be computed as a product of four basic transformations, a $z-$axis rotation (Rot_{z,θ_i}), a $z-$axis translational ($Trans_{z,d_i}$), a $x-$axis translational ($Trans_{x,a_i}$) and a $x-$axis rotation (Rot_{x,α_i}).

$$\begin{aligned}\mathbf{A}_i = \;& Rot_{z,\theta_i}\, Trans_{z,d_i}\, Trans_{x,a_i}\, Rot_{x,\alpha_i} \\ & \begin{bmatrix} C_{\theta_i} & -S_{\theta_i}C_{\alpha_i} & S_{\theta_i}S_{\alpha_i} & a_iC_{\theta_i} \\ S_{\theta_i} & C_{\theta_i}C_{\alpha_i} & -C_{\theta_i}S_{\alpha_i} & a_iS_{\theta_i} \\ 0 & S_{\alpha_i} & C_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}\end{aligned} \tag{A.3}$$

where $C_\phi = \cos(\phi)$ and $S_\phi = \sin(\phi)$. The four geometric parameter θ_i, d_i, a_i, α_i are associated with link i and joint i. This paramaters are generally called **link length**, **link twist**, **link offset**, and **joint angle**, respectively. Three of the above four geometric parameters are constant for a given link, while the fourth parameter, θ_i for a revolute joint or d_i for a prismatic joint, is the joint variable, thus, the matrix $\mathbf{A}_i$ is a function of a single variable.

Using the previous parametrization, it is possible to express the position and orientation of the end-point of link j with respect to an inertial or base frame i by a three-dimensional vector $\mathbf{o}_j^i$ (which gives the coordinates of the origin of the end-point frame with respect to the base frame) and the 3×3 rotation matrix $\mathbf{R}_j^i$, which are defined in the homogeneous transformation matrix as:

$$\mathbf{T}_j^i = \begin{bmatrix} \mathbf{R}_j^i & \mathbf{o}_j^i \\ \mathbf{0}^T & 1 \end{bmatrix} \tag{A.4}$$

Let us summarize the subsection. The forward kinematics of a serial manipulator can be computed by applying the Denavit Hartenberg representation, which allows computing the orientation and orientation of the end-effector of a serial mechanism, systematically. This representation uses four geometric parameters per joint, where three of them are constant and the fourth is usually variable. Thus, for a serial mechanism with n-DOF there are $3n$ constant geometric parameters. As a convention, the DH parameters are presented in a table called DH, for instance Table A.1.

Table A.1: Example of an DH table.

θ_i	d_i	a_i	α_i

The DH representation is used to describe the different serial manipulators subject to optimization in Section 7. This allow us to present their geometric parameters.

A.1.2 Kinematics

As described in the previous section, the forward kinematics define a function f relating the joint space position and the Cartesian space position in translation and orientation. The velocity relationships are then determined by the Jacobian of the function, as introduced in Section 3.2. In this section, we briefly present the derivation of the Jacobian for an $n-$link serial manipulator. The Jacobian represents the instantaneous transformation between the $n-$vector of joint velocities and the $6-$vector consisting of the linear and angular velocities of the end-effector, which is expressed as, $\dot{\mathbf{x}} = \mathbf{J}\dot{\boldsymbol{\theta}}$. This Jacobian is then a $6 \times n$ matrix.

Let us recall subsection 3.2, where we seek expressions of the form

$$\begin{bmatrix} \mathbf{v}_n^0 \\ \boldsymbol{\omega}_n^0 \end{bmatrix} = \begin{bmatrix} \mathbf{J}_v \\ \mathbf{J}_\omega \end{bmatrix} \dot{\boldsymbol{\theta}} = \mathbf{J}_n^0 \dot{\boldsymbol{\theta}} \tag{A.5}$$

where $\mathbf{v}_n^0$ denote the linear velocity of the end effector, $\boldsymbol{\omega}_n^0$ define the angular velocity vector of the end-effector, $\mathbf{J}_v = [\mathbf{J}_{v_1} \cdots \mathbf{J}_{v_n}]$ and $\mathbf{J}_\omega = [\mathbf{J}_{\omega_1} \cdots \mathbf{J}_{\omega_n}]$ are the $3 \times n$ linear and angular Jacobian matrices, respectively. Finally, $\mathbf{J}_n^0$ is the $6 \times n$ manipulator Jacobian matrix, where n is the number of joints.

Let us derive the linear and angular Jacobian separately for two specific type of joints, prismatic and revolute joints. Firstly, we derive the linear velocities for both cases as:

- **Case 1: Prismatic Joint**

  If joint i is prismatic, then it generates a pure translation to the end-effector. The position of the joint with respect to the base frame is given by $\mathbf{o}_n^0 = \mathbf{R}_i^0 \mathbf{o}_n^i + \mathbf{R}_{i-1}^0 \mathbf{o}_i^{i-1} + \mathbf{o}_{i-1}^0$. If only joint i is allowed to move, then both of $\mathbf{o}_n^i$ and $\mathbf{o}_{i-1}^0$ are constant. Furthermore, if joint i is prismatic, then the rotation matrix $\mathbf{R}_{i-1}^0$ is also constant. Thus, differentiation of $\mathbf{o}_n^0$ gives

$$\frac{\partial \mathbf{o}_n^0}{\partial t} = \dot{d}_i \mathbf{z}_{i-1}^0 = \mathbf{J}_{v_i} \dot{d}_i \tag{A.6}$$

  where d_i is the joint variable for primatic joint i

- **Case 2: Revolute Joint**

  Starting from the previous position equation, $\mathbf{o}_n^0 = \mathbf{R}_i^0 \mathbf{o}_n^i + \mathbf{R}_{i-1}^0 \mathbf{o}_i^{i-1} + \mathbf{o}_{i-1}^0$, $\mathbf{R}_i^0$ is not constant with respect to θ_i.

$$\frac{\partial \mathbf{o}_n^0}{\partial t} = \dot{\theta}_i \mathbf{z}_{i-1}^0 \times (\mathbf{o}_n^0 - \mathbf{o}_{i-1}^0) = \mathbf{J}_{v_i} \dot{\theta}_i \tag{A.7}$$

  where θ_i is the joint variable for revolute joint i. We can addressed $\mathbf{o}_n - \mathbf{o}_{i-1} = r$ and $z_{i1} = \omega$, therefore, a familiar expression is shown as $v = \omega \times r$.

Let us derivate now the angular velocities for both cases. To accomplish that, we must use a property of addition of angular velocities, where $\boldsymbol{\omega}_n^0 = \boldsymbol{\omega}_1^0 + \mathbf{R}_1^0 \boldsymbol{\omega}_2^1 + \cdots + \mathbf{R}_n^0 \boldsymbol{\omega}_n^{n-1}$. This means that we can determine the angular velocity of the end-effector relative to the base by expressing the angular velocity contributed by each joint in the orientation of the base frame and then summing them. The angular velocity of joint i expressed in the frame $i-1$ is given by $\boldsymbol{\omega}_i^{i-1} = \theta_i \mathbf{k}$, where $\mathbf{k}$ is the unit coordinate vector $[0 \quad 0 \quad 1]^T$. For both prismatic and revolute joints, the angular velocity can be derived as:

- **Case 1: Prismatic Joint**

  If the i^{th} joint is prismatic, then the motion of frame i relative to frame $i-1$ is a translation and $\boldsymbol{\omega}_i^{i-1} = \mathbf{0}$. Thus, the angular velocity of the end-effector does not depend on the joint variable θ_i, in this case d_i.

- **Case 2: Revolute Joint**

  As mentioned in the previous case, the angular velocity is determined as the contribution of each joint in the orientation of the base frame. Therefore, the overall angular velocity for a revolute joint, $\boldsymbol{\omega}_i^0$, in the base frame is determined by

$$\boldsymbol{\omega}_i^0 \quad = z_{i-1}^0 \dot{\boldsymbol{\theta}} \quad = \mathbf{J}_{\omega_\mathrm{i}} \dot{\boldsymbol{\theta}} \tag{A.8}$$

where $\dot{\boldsymbol{\theta}} = (\dot{\theta}_1, \ldots, \dot{\theta}_n)$. Combining both Jacobians as in Eq. A.5, we can see that for each joint the manipulator Jacobian is given by:

$$\mathbf{J}_i^{(R)} = \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{o}_n - \mathbf{o}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix}; \qquad \mathbf{J}_i^{(P)} = \begin{bmatrix} \mathbf{z}_{i-1} \\ \mathbf{0} \end{bmatrix} \tag{A.9}$$

where $\mathbf{J}_i^{(R)}$ is the i^{th} column of the manipulator Jacobian matrix for a revolute joint and $\mathbf{J}_i^{(P)}$ is the i^{th} column of the manipulator Jacobian matrix for a prismatic joint.

A.1.3 Anthropomorphic manipulator kinematic

The optimization problem presented in Section 7.1 are addresses for a complex serial robotic manipulator, particularly an anthropomorphic robot arm. In this section, we present the anthropomorphic serial manipulator with 6−DOF, as well as its geometric parameters. An anthropomorphic manipulator is a type of complex mechanism, with similar functions to a human arm, where it gets its name (from the greek *ánthrōpos*,"human" and *morphē*, "form"). It is widely used in the industry, from assembly operations, diecasting, fettling machines, to gas welding, arc welding and spray paint. It is a robot whose arm has at least three revolute joints and 3 more in the wrist. Fig. A.1 shows the schematic architecture of a 6−DOF anthropomorphic manipulator. As it can be seen, the manipulator can be divided into body, from the base to the third link, and wrist, from the fourth joint to the end-effector. The geometric DH parameters are shown in Table A.2, where the DH variable are marked with an *. In this case, all the DH parameters are angles due to all joint are revolute types.

As can be seen in Table A.2, we can group the constant geometric parameters in two, the morphologic parameters, defined by the type of manipulator and its architecture, and

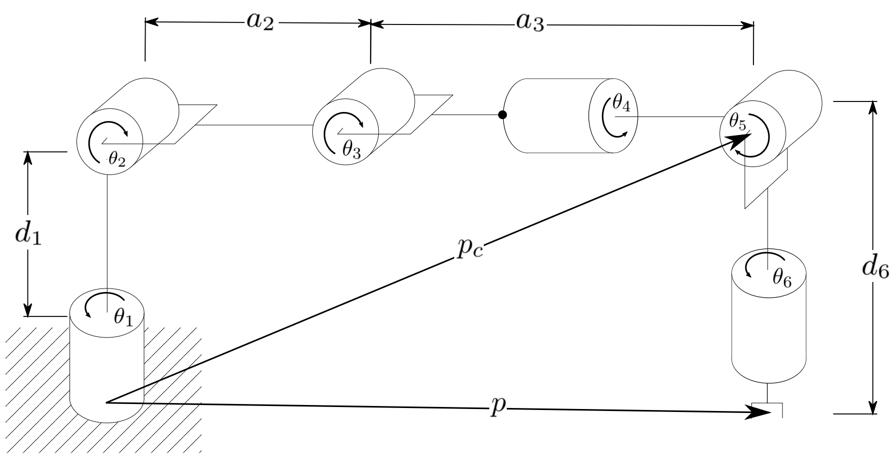


FIGURE A.1: Schematic of the architecture of an anthropomorphic robot.

TABLE A.2: DH table for a 6−DOF anthropomotphic manipulator.

θ_i	d_i	a_i	α_i
θ_1^*	d_1	0	$\pi/2$
θ_2^*	0	a_2	0
θ_3^*	0	a_3	$\pi/2$
θ_4^*	0	0	$-\pi/2$
θ_5^*	0	0	$\pi/2$
θ_6^*	d_6	0	0

the design parameters. For this type of manipulator, we have four design parameters that correspond to lengths d_1, a_2, a_3 and a_6. The forward kinematics and velocity kinematics of the manipulator are given as shown in Appendix A.1.1 and A.1.2.

A.1.4 Two-link planar manipulator kinematic

The optimization problems presented in Section 7.3 are addressed for a simple robotic manipulator, a two-link planar manipulator. In this section, we briefly present the kinematic modeling of this manipulator and its geometric parameters. The two-link planar manipulator is a simple mechanism, usually used as a test benchmark for a dynamic control. As shown in the schematic in Fig. A.2, the mechanism is made of two links united by an actuated rotational joint and fixed to the base by another actuated rotational joint. Its geometric parameters are shown in Table A.3, where the joint variables are marked with an *. As in the previous manipulator, the constant parameters can be grouped in two, This manipulator has two design parameters. The forward kinematic of the manipulator can be computed as shown in Appendix A.1.1,

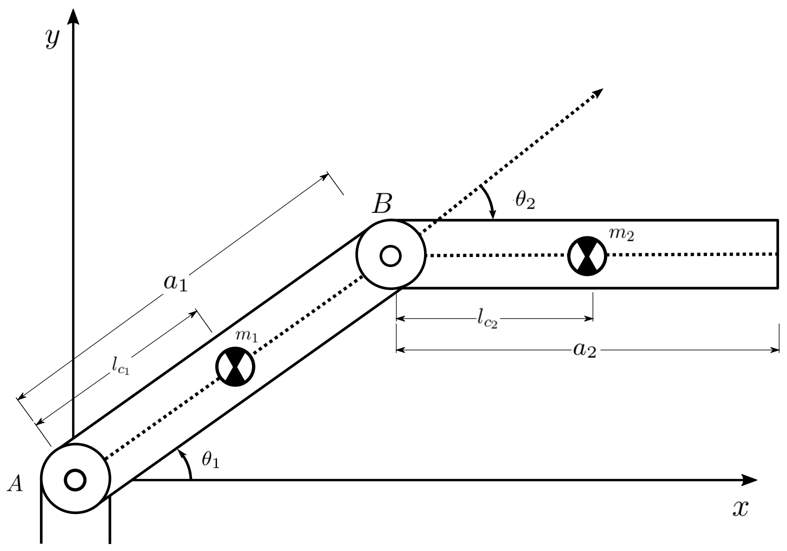


FIGURE A.2: Schematic of the architecture of a two-link planar manipulator.

TABLE A.3: DH table for a 2−DOF two-link planar manipulatror.

θ_i	0	a_i	α_i
θ_1^*	0	a_1	0
θ_2^*	0	a_2	0

but a direct formulation is presented as:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_1 C_{\theta_1} + a_2 C_{\theta_1+\theta_2} \\ a_1 S_{\theta_1} + a_2 S_{\theta_1+\theta_2} \end{bmatrix} \tag{A.10}$$

The manipulator Jacobin matrix is:

$$\mathbf{J} = \begin{bmatrix} -(a_1 S_{\theta_1} + a_2 S_{\theta_1+\theta_2}) & -a_2 S_{\theta_1+\theta_2} \\ a_1 C_{\theta_1} + a_2 C_{\theta_1+\theta_2} & a_2 C_{\theta_1+\theta_2} \end{bmatrix} \tag{A.11}$$

A.2 Serial manipulator dynamic

The optimization problems presented in Sections 7.3.1, 7.3.2, 7.3.3 and 7.3.4 are addressed from a dynamic modeling. In this section, we summarize a general manipulator dynamics based on the Lagrange formulation [26, 27]. For it, let us recall the Section 3.3, more specifically Eq. (3.5), where the dynamic of a kinematically complex mechanism can be modeled as a differential equation.

$$\mathbf{B}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\dot{\boldsymbol{\theta}}, \boldsymbol{\theta})\dot{\boldsymbol{\theta}} + \mathbf{F}_v\dot{\boldsymbol{\theta}} + \mathbf{F}_s\mathbf{sgn}(\dot{\boldsymbol{\theta}}) + \mathbf{G}(\boldsymbol{\theta}) = \tau - \mathbf{J}^T(\boldsymbol{\theta})\mathbf{h}_e \tag{A.12}$$

where the internal forces are represented by the inertial matrix $\mathbf{B}$, the Coriolis matrix $\mathbf{C}$ and the gravitational vector $\mathbf{G}$. Thus, we present the general procedure to compute the components of the internal forces. It is worth mentioning that in the optimization problems, we do not consider that the robot is subject to external forces. Thus, we do not present the mathematical procedure to compute the components of the external forces $\mathbf{F}_v$, $\mathbf{F}_s$ and $\mathbf{h}_e$.

As introduced in Section 3.3, the base of the Lagrange formulation is the equation $\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\theta}}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}\right)^T = \xi$, where ξ is the generalized force associated with the generalized coordinates $\boldsymbol{\theta}$, $\mathcal{T}$ and $\mathcal{U}$ denote the total kinetic energy and potential energy of the mechanism, respectively. Let us introduce the computation of the total kinetic energy for serial manipulators as the the sum of the contributions related to the motion of each actuated joint as $\mathcal{T} = \sum_{i=1}^{n} \mathcal{T}_i$, where $\mathcal{T}_i$ is the kinetic energy of link i given as:

$$\mathcal{T}_i = \frac{1}{2}\int_{V_{l_i}} \dot{\mathbf{p}}_i^{*T}\dot{\mathbf{p}}_i^{*}\rho dV \tag{A.13}$$

where $\dot{\mathbf{p}}_i^*$ is the linear velocity vector and ρ is the density of the element of volume dV; V_{l_i} is the total volume of link i. Let us assume that for link i, the position vector of any segment of the link $\mathbf{p}_i^*$ and the position vector of its center of mass $\mathbf{p}_{l_i}$ are expressed in the same base frame. We can interpret $\mathbf{r}_i = \mathbf{p}_i^* - \mathbf{p}_{l_i}$ as the distance between the segment of the link and the center of mass, which can be computed as

$$\mathbf{p}_{l_i} = \frac{1}{m_{l_i}}\int_{V_{l_i}} \mathbf{p}_i^{*}\rho dV \tag{A.14}$$

where m_{l_i} is the link mass. As consequence, the link point velocity can be expressed as $\mathbf{p}_i^* = \dot{\mathbf{p}}_{l_i} + \mathbf{S}(\boldsymbol{\omega}_i)\mathbf{r}_i$, where $\dot{\mathbf{p}}_{l_i}$ is the linear velocity of the center of mass, $\boldsymbol{\omega}_i$ is the angular velocity of the i^{th} link and $\mathbf{S}(\boldsymbol{\omega}_i)\mathbf{r}_i$ is the inertial contribution of the i^{th} link to the angular velocity. By substituing the position expression in Eq. (A.14) into Eq. (A.13), it can be recognized that the kinetic energy of each link is formed by the contributions of:

1. **Translational**

$$\frac{1}{2}\int_{V_{l_i}} \dot{\mathbf{p}}_i^{*T}\dot{\mathbf{p}}_i^{*}\rho dV = \frac{1}{2}m_{l_i}\dot{\mathbf{p}}_{l_i}{}^T\dot{\mathbf{p}}_{l_i} \tag{A.15}$$

2. **Rotational**

$$\frac{1}{2}\int_{V_{l_i}} \mathbf{r}_i^T\mathbf{S}^T(\boldsymbol{\omega}_i)\mathbf{S}(\boldsymbol{\omega}_i)\mathbf{r}_i\rho dV = \frac{1}{2}\boldsymbol{\omega}_i^T\mathbf{I}_{l_i}\boldsymbol{\omega}_i \tag{A.16}$$

where $\mathbf{I}_{l_i}$ the inertial tensor of the i^{th} link relative to the center of mass when expressed in the base frame, which depends on the manipulator configuration. If the angular framework of link i is expressed with reference to a frame attached to the link $\boldsymbol{\omega}_i^i = \mathbf{R}^T\mathbf{w}_i$, where $\mathbf{R}_i$ is the rotation matrix from link i frame to the base frame. When referred to the link frame, the inertial tensor is constant, this simplifies the computation of the inertial tensor as

$$\mathbf{I}_{l_i} = \mathbf{R}_i\mathbf{I}_{l_i}^i\mathbf{R}_i^T \tag{A.17}$$

It is necessary to express the kinetic energy as a function of the generalized coordinates of the system. Recalling Eq. (A.5), we can use the geometric method for Jacobian computation, and compute the velocities for the intermediate links other than the end-effector, yielding

$$\begin{bmatrix} \dot{\mathbf{p}}_{l_i} \\ \boldsymbol{\omega}_i \end{bmatrix} = \begin{bmatrix} \mathbf{J_{v_1}}^{(l_i)}\dot{\boldsymbol{\theta}}_1 + \cdots + \mathbf{J_{v_i}}^{(l_i)}\dot{\theta}_i \\ \mathbf{J}_{\omega_1}{}^{(l_i)}\dot{\boldsymbol{\theta}}_1 + \cdots + \mathbf{J}_{\omega_\mathbf{i}}{}^{(l_i)}\dot{\boldsymbol{\theta}}_i \end{bmatrix} = \begin{bmatrix} \mathbf{J_v}^{(l_i)}\dot{\boldsymbol{\theta}} \\ \mathbf{J}_\omega{}^{(l_i)}\dot{\boldsymbol{\theta}} \end{bmatrix} \tag{A.18}$$

By adding the traslational and rotational contributions from Eq. (A.15) and Eq. A.16, and expressed as a function of the generalized coordinates, the kinetic energy of link i is

$$\mathcal{T}_i = \frac{1}{2}m_{l_i}\dot{\boldsymbol{\theta}}^T\mathbf{J}_P^{(li)^T}\mathbf{J}_P^{(li)}\dot{\boldsymbol{\theta}} + \frac{1}{2}\dot{\boldsymbol{\theta}}^T\mathbf{J}_O^{(li)^T}\mathbf{R}_i\mathbf{I}_{l_i}^i\mathbf{R}_i^T\mathbf{J}_P^{(li)}\dot{\boldsymbol{\theta}} \tag{A.19}$$

Once again, by adding the contributions relative to the links, the total kinetic energy of the manipulator with actuators is given by the quadratic form

$$\mathcal{T} = \frac{1}{2}\dot{\boldsymbol{\theta}}^T\mathbf{B}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{A.20}$$

where $\mathbf{B}(\boldsymbol{\theta})$ is the inertial matrix of the manipulator and is defined as:

$$\mathbf{B}(\theta) = \sum_{i=1}^{n}\frac{1}{2}m_{l_i}\mathbf{J}_P^{(li)^T}\mathbf{J}_P^{(li)} + \frac{1}{2}\mathbf{J}_O^{(li)^T}\mathbf{R}_i\mathbf{I}_{l_i}^i\mathbf{R}_i^T\mathbf{J}_P^{(li)} \tag{A.21}$$

As done for kinetic energy, let us introduce now the computation of the total potential energy for serial manipulators as the sum of the contributions related to the stored energy of each actuated joint as $\mathcal{U} = \sum_{i=1}^n \mathcal{U}_i$, where $\mathcal{U}_i$ is the potential energy of link i given as:

$$\mathcal{U}_i = -\int_{V_{li}} \mathbf{g}_0^T\mathbf{p}_i^*\rho dV = -m_{l_i}\mathbf{g}_0^T\mathbf{p}_{l_i} \tag{A.22}$$

where $\mathbf{g}_0$ is the gravity acceleration vector, expressed in the base frame, which is exerted over the center of mass.

Once we have derived the total kinetic and potential energy for the serial manipulators, applying the first kind Lagrange equation, as $\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\theta}}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}\right)^T = \xi$, the Coriolis matrix and the gravitational vector are computed as follows:

- **Coriolis Matrix $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$**

$$\mathbf{C}(\theta) = \dot{\mathbf{B}}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} - \frac{1}{2}\left(\frac{\partial}{\partial \boldsymbol{\theta}}\left(\dot{\boldsymbol{\theta}}^T \mathbf{B}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}\right)\right) \tag{A.23}$$

- **Gravitational Vector $\mathbf{G}(\boldsymbol{\theta})$**

$$\mathbf{G}(\theta) = -\sum_{j=1}^{n} m_{l_i} \mathbf{g}_0^T \mathbf{J}_{P_i}^{(l_i)}(\boldsymbol{\theta}) \tag{A.24}$$

As can be noticed, the Coriolis matrix need the closed form differentiable function of the inertia matrix. But the inertia matrix depends on the morphology of the manipulator. Thus, in general, the Coriolis matrix can be computed by using a numerical differentiation.

A.2.1 Two-link planar manipulator dynamic

In Sections 7.3.1, 7.3.2 and 7.3.3, a two-link planar manipulator is optimized, applying a dynamic model. In this section, the particular expressions for the inertia matrix, Coriolis matrix and gravitational vector for the two-link planar manipulator are given. Let us recall Fig. A.2.

To derive the dynamic model of a two-link planar manipulator, let us reference to [53], where, once obtained the joints position and velocity, the dynamic model can be seen as dependent on the inertia matrix, Coriolis matrix, and gravitation vector. They are given as:

- **Inertia matrix $\mathbf{B}(\boldsymbol{\theta})$**

$$\mathbf{B}(\boldsymbol{\theta}) = \begin{bmatrix} k_{11} + k_{12}C_{\theta_1} & k_{31} + k_{32}C_{\theta_1} \\ k_{31} + k_{32}C_{\theta_1} & 2k_2 \end{bmatrix} \tag{A.25}$$

- **Coriolis matrix $\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$**

$$\mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \begin{bmatrix} k_{12} S_{\theta_2} \dot{\theta}_1 & k_{32} \dot{\theta}_2 \\ 0.5 k_{12} S_{\theta_2} \dot{\theta}_2 & 0 \end{bmatrix} \tag{A.26}$$

- **Gravitational vector $\mathbf{G}(\boldsymbol{\theta})$**

$$\mathbf{G}(\boldsymbol{\theta}) = g \begin{bmatrix} p_1 C_{\theta_1} + p_2 C_{\theta_1+\theta_2} \\ p_2 C_{\theta_1+\theta_2} \end{bmatrix} \tag{A.27}$$

with

$$\begin{aligned} k_{11} &= I_1 + I_2 + m_2(l_1 + l_{c_2}) \\ k_{12} &= 2 m_2 l_1 l_{c_2} \\ k_2 &= 0.5 \left(I_2 + m_2 l_{c_2}^2 \right) \\ k_{31} &= I_2 + m_2 l_{c_2}^2 \\ k_{32} &= m_2 l_1 l_6 \\ p_1 &= (m_2 l_1 + m_1 l_{c_1}) \\ p_2 &= m_1 l_{c_1} \end{aligned}$$

where m_1 and m_2 are the masses of links 1 and 2; l_1 is the distance between the joint centres A and B; θ_1 and $\dot{\theta}_1$ are the angular position and velocity of link 1 relative to the base, respectively; θ_2 and $\dot{\theta}_2$ is the angular position and velocity of link 2 relative to link 1, respectively; I_1 is the axial moment of inertia of link 1 relative to A; l_{c_1} is the distance between the centre of mass c_1 of link 1 and joint centre A; I_2 is the axial moment of inertia of link 2 relative to the centre of mass c_2 of link 2; l_{c_2} is the distance between the centre of mass c_2 of link 2 and joint centre B; g is the gravitational acceleration.

A.2.2 Counterweigth two-link planar manipulator dynamic

In Section 7.3.4, the optimization of a two-link planar manipulator with counterweigth is presented. Thus, let us consider the input torque due to the counterweights. For this, we can rewrite the inertial matrix as $\mathcal{B}(\boldsymbol{\theta}) = \mathbf{B}(\boldsymbol{\theta}) + \mathbf{B}_w(\boldsymbol{\theta})$, the Coriolis matrix as $\mathcal{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) + \mathbf{C}_w(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$, and the gravitational vector as $\mathcal{G}(\boldsymbol{\theta}) = \mathbf{G}(\boldsymbol{\theta}) + \mathbf{G}_w(\boldsymbol{\theta})$, where $\mathbf{B}_w$, $\mathbf{C}_w$ and $\mathbf{G}_w$ are the momentum contributions of the counterweights; $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{G}$ are the the momentum contributions of the links 1 and 2; and $\mathcal{B}$, $\mathcal{C}$ and $\mathcal{G}$ are the general momentum parameters of the manipulator. The expressions for the contributions given by the counterweigths are as follows:

- **Counterweigth inertia matrix $\mathbf{B}_w(\boldsymbol{\theta})$**

$$\mathbf{B}_w(\boldsymbol{\theta}) = \begin{bmatrix} k_{w_{11}} + k_{w_{12}} C_{\theta_1} & k_{w_{31}} + k_{w_{32}} C_{\theta_1} \\ k_{w_{31}} + k_{w_{32}} C_{\theta_1} & k_{w_2} \end{bmatrix} \tag{A.28}$$

- **Counterweigth Coriolis matrix $\mathbf{C}_w(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$**

$$\mathbf{C}_w(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) = \begin{bmatrix} k_{w_{12}} S_{\theta_1} \dot{\theta}_1 & k_{w_{32}} \dot{\theta}_2 \\ 0.5 k_{w_{12}} S_{\theta_1} \dot{\theta}_1 & 0 \end{bmatrix} \tag{A.29}$$

- **Counterweigth gravitational vector $\mathbf{G}_w(\boldsymbol{\theta})$**

$$\mathbf{G}_w(\boldsymbol{\theta}) = g \begin{bmatrix} p_{w_1} C_{\theta_1} + p_{w_2} C_{\theta_1+\theta_2} \\ p_{w_2} C_{\theta_1+\theta_2} \end{bmatrix} \tag{A.30}$$

with

$$\begin{aligned} k_{w_{11}} &= m_{w_2} l_{w_1} + m_{w_2}(l_1 + l_{w_2}) \\ k_{w_{12}} &= -2 m_{w_2} l_1 l_{w_2} \\ k_{w_2} &= 0.5 m_{w_2} l_{w_2} l_{w_2} \\ k_{w_{31}} &= m_{w_2} l_{w_2} l_{w_2} \\ k_{w_{32}} &= -m_{w_2} l_1 l_{w_2} \\ p_{w_1} &= (m_{w_1} l_{w_1} + m_{w_2} l_1) \\ p_{w_2} &= m_{w_2} l_{w_2} \end{aligned}$$

where m_{w_1} and m_{w_2} are the masses of the counterweights; l_{w_1} is the rotation radius of the centre of mass of the counterweight with respect to A; l_{w_2} is the rotation radius of the centre of mass of the counterweight with respect tp B.

Appendix B

Parallel manipulator

In this chapter, the kinematics model for the Delta parallel manipulator is addressed. A parallel robot consists of a mobile platform and a fixed platform as a base, connected by several *arms*. This kind of mechanisms possesses high rigidity, load capacity, precision, structural stiffness, velocity and acceleration since the end-effector is linked to the mobile plate in several points. Parallel manipulators can be classified into two fundamental categories, namely spatial and planar manipulators. The first category considers parallel manipulators that can translate and rotate in the three-dimensional space. The second category consist of parallel manipulators that can translate along the x and y-axes, and rotate around the z-axis, only. Planar parallel manipulators are increasingly being used in industry for micro or nanopositioning applications. However, the mathematical modeling of dynamics and even kinematics of planar parallel manipulators is more difficult than their serial counterparts. Therefore, selection of an efficient kinematic modeling convention is important to simplify the complexity of planar parallel manipulators modeling. In this section, the kinematic model of a 3-DOF RPR Delta parallel manipulator is derived. The reader can refer to [6, 30, 54] for more details.

B.1 Delta parallel manipulator

The Delta robot was introduced by Clavel [6] as a 3-DOF parallel robot, dedicated to high speeds applications. Consisting of three kinematic chains, that connects the base with the end-effector, each kinematic chain is driven by a revolute joint, located at the base. The key design feature is the parallelogram link in each arm, that allows maintaining the orientation of the end-effector. Fig. B.1 shows a 3D structure of a parallel Delta manipulator. The mechanism consists of a base, a moving platform, and three identical kinematic chains. All sub-chains have a common $\underline{\mathcal{R}}\mathcal{R}\mathcal{P}_a\mathcal{R}$, as shown in

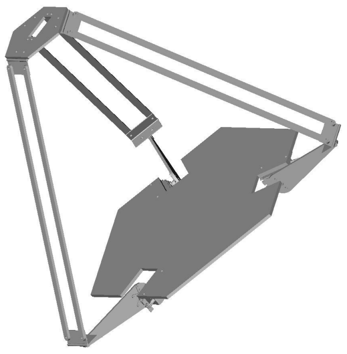


FIGURE B.1: 3D structure of a parallel Delta manipulator.

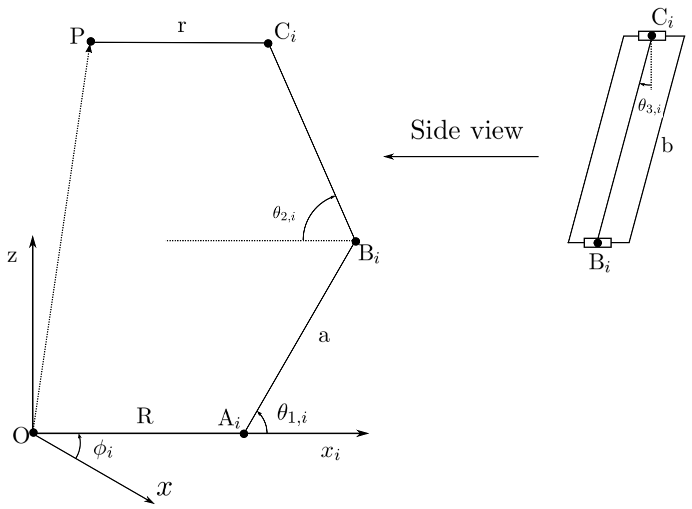


FIGURE B.2: Axis angles on the $i - th$ *arm* over point A_i.

Fig. B.2, topology from the base to the moving platform, where $\mathcal{R}$ denotes a revolute joint in C_i and B_i, $\mathcal{P}_a$ a parallelogram from C_i to B_i, and $\underline{R}$ an actuated revolute joint in A_i. Three actuated joints on the base are arranged symmetrically at the three vertices of an equilateral triangle. There are the three passive joints on the moving platform. The kinematic parameters are depicted in Figure B.2, where a denotes the length of arms $\overline{A_iB_i}$, b the length of the parallelogram $\overline{B_iC_i}$, $R = \overline{OA_i}$ and $r = \overline{PC_i}$, with O and P being the centers of the base and the moving platform, respectively.

B.1.1 Inverse Kinematics

The coordinate-based framework the a parallel manipulator angles of the $i - th$ *arm* is shown in Fig. B.2, where $\mathbf{p}$ is the vector from the center of the fixed base to the center of the mobile base, $\theta_{1,i}$ is the angle between the x_i-axis to link $\overline{OA_i}$, $\theta_{2,i}$ is the complementary angle between $\overline{OA_i}$ and $\overline{A_iB_i}$, $\theta_{3,i}$ is the angle between the y_i-axis an the the link $\overline{B_iC_i}$. Let us move the origin reference framework to the $A_{i_{x_i,y_i,z_i}}$ framework, the closed loop for each *arm* is defined as:

$$\mathbf{p} - \mathbf{r} = \mathbf{a}_i + \mathbf{b}_i \tag{B.1}$$

where $\mathbf{r}$ is the difference between the center of the center of the mobile base and the fixed based, relative to the reference framework. $\mathbf{a}_i$ is the distance from point A_i to B_i, relative to the reference framework, and finally, $\mathbf{b}_i$ is the distance from point B_i to C_i.

$$\begin{bmatrix} \cos(\phi_i) & \sin(\phi_i) & 0 \\ -\sin(\phi_i) & \cos(\phi_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} - \begin{bmatrix} r - R \\ 0 \\ 0 \end{bmatrix} = a \begin{bmatrix} \cos(\theta_{1,i}) \\ 0 \\ \sin(\theta_{1,i}) \end{bmatrix} + b \begin{bmatrix} \sin(\theta_{3,i})\cos(\theta_{1,i}+\theta_{2,i}) \\ \cos(\theta_{3,i}) \\ \sin(\theta_{3,i})\sin(\theta_{1,i}+\theta_{2,i}) \end{bmatrix} \tag{B.2}$$

Angle $\theta_{3,i}$ can be computed as in (B.3), solving the left side of the equation (B.2), where $c_{x,i} = P_x\cos(\phi_i) + P_y\sin(\phi_i)$, $c_{y,i} = -P_x\sin(\phi_i) + P_y\cos(\phi_i)$ and $c_{x,i} = P_z$

$$\theta_{3,i} = \cos^{-1}\left(\frac{c_{y,i}}{b}\right) \tag{B.3}$$

Angle $\theta_{2,i}$ can be computed as the adjacent angle to the triangle $\overline{A_iB_iC_i}$, resulting in the Eq. (B.4).

$$\theta_{2,i} = \cos^{-1}\left(\frac{c_{x,i}^2 + c_{y,i}^2 + c_{z,i}^2 - a^2 - b^2\sin^2(\theta_{3,i})}{2ab\sin(\theta_{3,i})}\right) \tag{B.4}$$

Finally, $\theta_{1,i}$ can be computed from Eq. (B.2), where the x_i and z_i axes are function of $\theta_{1,i}$. Applying the trigonometric identities of the sum and difference of angles, and by isolating the trigonometric functions in function of $\theta_{1,i}$, we have:

$$\begin{bmatrix} c_{x,i} \\ c_{z,i} \end{bmatrix} = \begin{bmatrix} a - b\sin(\theta_{3,i})\cos(\theta_{2,i}) & -b\sin(\theta_{3,i})\sin(\theta_{2,i}) \\ b\sin(\theta_{3,i})\sin(\theta_{2,i}) & a - b\sin(\theta_{3,i})\cos(\theta_{2,i}) \end{bmatrix} \begin{bmatrix} \cos(\theta_{1,i}) \\ \sin(\theta_{1,i}) \end{bmatrix} \tag{B.5}$$

Solving the corresponding equations system, $\theta_{1,i}$ can be computed through the Eq. (B.6)

$$\theta_{1,i} = \tan^{-1}\left(\frac{\frac{c_{z,i}}{c_{x,i}}\frac{n}{m} - 1}{\frac{n}{m} + \frac{c_{z,i}}{c_{x,i}}}\right) \tag{B.6}$$

where $n = a - b\sin(\theta_{3,i})\cos(\theta_{2,i})$ y $m = b\sin(\theta_{3,i})\sin(\theta_{2,i})$

B.1.2 Jacobian matrix

In this section, where the relation of the angular velocities and the end-effector velocity for a parallel Delta manipulator is addressed. We are looking for expression in the form:

$$\mathbf{J}_v\dot{\mathbf{x}} = \mathbf{J}_\theta\dot{\boldsymbol{\theta}} \tag{B.7}$$

where $\mathbf{J}_v \in \mathbb{R}^{3\times3}$ and $\mathbf{J}_\theta \in \mathcal{R}^{3\times3}$ are the cartesian and angular jacobian, respectively. $\dot{\mathbf{x}} \in \mathbb{R}^3$ and $\dot{\boldsymbol{\theta}} \in \mathbb{R}^3$ are the cartesian velocities of the end- effector and angular velocities from the actuated joints, respectively, as in [30]. Differentiating (B.1) with respect of time and taking into account that the vector $\mathbf{r}$ is constant, we have:

$$\dot{\mathbf{p}} \quad = \quad \dot{\mathbf{a}}_i + \dot{\mathbf{b}}_i \tag{B.8}$$

In this expression, each point on the moving platform moves exactly in the same direction. The linear velocities on the right side of the equation can be changed into the angular velocities using a well known identity, where $\dot{\mathbf{x}} = \boldsymbol{\omega}\times\mathbf{x}$, as $\dot{\mathbf{p}} = \boldsymbol{\omega}_{a,i}\times\mathbf{a}_i+\boldsymbol{\omega}_{b,i}\times\mathbf{b}_i$, where $\boldsymbol{\omega}_{a,i}$ and $\boldsymbol{\omega}_{b,i}$ are the angular velocities of the actuated and passive joint, respectively, at point A_i and B_i of the i^{th} arm. Eq. (B.8) can be written as:

$$(\dot{\mathbf{p}} \quad = \quad \boldsymbol{\omega}_{a,i} \times \mathbf{a}_i + \boldsymbol{\omega}_{b,i} \times \mathbf{b}_i) \cdot \hat{b}_i$$

The vector $\omega_{b,i}$ introduces dependencies between variables $\dot{\theta}_{2,i}$ and $\dot{\theta}_{3,i}$. However, it can be overcome as follows. The term $\boldsymbol{\omega}_{b,i} \times \mathbf{b}_i$ can be eliminated due to the dot product, taking into account that the linear velocity is orthogonal to the radius generated by the angular velocity, i.e., $\dot{\mathbf{x}} \cdot \hat{\mathbf{x}} = 0$. Then we have:

$$\hat{b}_i \cdot \dot{\mathbf{p}} \quad = \quad \hat{b}_i \cdot \boldsymbol{\omega}_{a,i} \times \mathbf{a}_i \tag{B.9}$$

Since $\hat{b}_i \cdot \boldsymbol{\omega}_{a,i} \times \mathbf{a}_i = 0$, let us solve the left side of the Eq. (B.9).

$$\begin{aligned}
\hat{b}_i \cdot \dot{\mathbf{p}} &= \begin{bmatrix} \sin(\theta_{3,i})\cos(\theta_{1,i}+\theta_{2,i}) \\ \cos(\theta_{3,i}) \\ \sin(\theta_{3,i})\sin(\theta_{1,i}+\theta_{2,i}) \end{bmatrix} \cdot \begin{bmatrix} \dot{P}_x\cos(\phi_i) + \dot{P}_y\sin(\phi_i) \\ -\dot{P}_x\sin(\phi_i) + \dot{P}_y\cos(\phi_i) \\ \dot{P}_z \end{bmatrix} \\
&= \dot{P}_x(\sin(\theta_{3,i})\cos(\theta_{1,i}+\theta_{2,i})\cos(\phi_i) - \cos(\theta_{3,i})\sin(\phi_i)) + \dots \\
&\quad \dots + \dot{P}_y(\sin(\theta_{3,i})\cos(\theta_{1,i}+\theta_{2,i})\sin(\phi_i) + \cos(\theta_{3,i})\cos(\phi_i)) + \dots \\
&\quad \dots + \dot{P}_z(\sin(\theta_{3,i})\sin(\theta_{1,i}+\theta_{2,i})) \\
&= \mathbf{J}_v\dot{\mathbf{x}}
\end{aligned}$$

with $\dot{\mathbf{x}} = \left[\dot{P}_x, \dot{P}_y, \dot{P}_z\right]$. As notice, each column of the angular Jacobian can be computed as:

$$\begin{bmatrix} j_{v_{i,x}} \\ j_{v_{i,y}} \\ j_{v_{i,z}} \end{bmatrix} = \begin{bmatrix} \sin(\theta_{3,i})\cos(\theta_{1,i}+\theta_{2,i})\cos(\phi_i) - \cos(\theta_{3,i})\sin(\phi_i) \\ \sin(\theta_{3,i})\cos(\theta_{1,i}+\theta_{2,i})\sin(\phi_i) + \cos(\theta_{3,i})\cos(\phi_i) \\ \sin(\theta_{3,i})\sin(\theta_{1,i}+\theta_{2,i}) \end{bmatrix} \tag{B.10}$$

Now, let us solve the right side of the Eq. (B.9). Because the link a_i moves in the plane x_i, z_i, its only velocity component is perpendicular the plane, given as $\boldsymbol{\omega} = \begin{bmatrix} 0 & \dot{\theta}_1 & 0 \end{bmatrix}^T$. Therefore $\boldsymbol{\omega}_{a,i} \times \mathbf{a_i} = a\left[\sin(\theta_{1,i}) \quad 0 \quad -\cos(\theta_{1,i})\right]^T \dot{\theta}_{1,i}$, thus

$$\hat{b_i} \cdot \boldsymbol{\omega}_{a,i} \times \mathbf{a}_i = \begin{bmatrix} \sin(\theta_{3,i})\cos(\theta_{1,i}+\theta_{2,i}) \\ \cos(\theta_{3,i}) \\ \sin(\theta_{3,i})\sin(\theta_{1,i}+\theta_{2,i}) \end{bmatrix} \cdot a \begin{bmatrix} \sin(\theta_{1,i}) \\ 0 \\ -\cos(\theta_{1,i}) \end{bmatrix} \dot{\theta}_{1,i} \tag{B.11}$$

We can see that $j_{\theta_i} = -a\sin(\theta_{3,i})\sin(\theta_{2,i})$, implying that $\mathbf{J}_v\dot{\mathbf{x}} = \mathbf{J}_\theta\dot{\boldsymbol{\theta}}$, where

$$\mathbf{J}_v = \begin{bmatrix} j_{v_{1,x}} & j_{v_{1,y}} & j_{v_{1,z}} \\ j_{v_{2,x}} & j_{v_{2,y}} & j_{v_{2,z}} \\ j_{v_{3,x}} & j_{v_{3,y}} & j_{v_{3,z}} \end{bmatrix} \; ; \; \mathbf{J}_\theta = diag\begin{bmatrix} j_{\theta_1} & j_{\theta_2} & j_{\theta_3} \end{bmatrix} \tag{B.12}$$

Recalling Section 3.2, the relationship between the Cartesian reference framework and the framework of each joint can be computed as $\dot{\mathbf{x}} = \mathbf{J}\dot{\boldsymbol{\theta}}$, where $\mathbf{J}$ is the Jacobian matrix. In this case, the Jacobian matrix is computed as $\mathbf{J} = \mathbf{J}_v^{-1}\mathbf{J}_\theta$. Because is easier and computationally more efficient to compute the inverse of a diagonal matrix, we suggest computing the inverted Jacobian as in Eq. (B.13), so it can be finally inverted, to get the Jacobian matrix.

$$\mathbf{J}^{-1} = \begin{bmatrix} \frac{j_{v_{1,x}}}{j_{\theta_1}} & \frac{j_{v_{1,y}}}{j_{\theta_1}} & \frac{j_{v_{1,z}}}{j_{\theta_1}} \\ \frac{j_{v_{2,x}}}{j_{\theta_2}} & \frac{j_{v_{2,y}}}{j_{\theta_2}} & \frac{j_{v_{2,z}}}{j_{\theta_2}} \\ \frac{j_{v_{3,x}}}{j_{\theta_3}} & \frac{j_{v_{3,y}}}{j_{\theta_3}} & \frac{j_{v_{3,z}}}{j_{\theta_3}} \end{bmatrix} \tag{B.13}$$

Appendix C

Controllers

As explained in the objective of the thesis, we addressed the problem of mechanism design in two senses: optimizing structural parameters and control parameters. In this section, we include a brief introduction of control techniques used in the experimental results, particularly in Sections 7.1.2, 7.2.2, 7.3.1, 7.3.2, 7.3.3 and 7.3.4. As mentioned in Section 3.4, *"The control is responsible for adjusting the velocities, forces or torques applied to the actuated joints of the mechanism, to perform the desired task. To change the behavior of the model, the control must be designed on the basis of a control law with feedbac"*. There are many control techniques and methodologies that can be applied to the control of a mechanism. In particular, the chosen control methods, as well as the manner in which they are implemented, can have a significant impact on the performance of the mechanism. Thus, in this chapter, the control schemes used in the implementation of the optimization problems are presented. For simplicity of the control technique and for comparative purposes with previous results, found in the literature, we firstly use a classical proportional-integral-derivative (PID) controller. Additionally, a variant of the PID controller, commonly applied in robotic systems, is also presented.

C.1 Proportional-integral-derivative controller

A proportional-integrative-derivative (PID) controller, is a generic control over a closed loop feedback, widely used in industry to control any kind of systems. The PID controller is described by three parameters: the proportional, integral and derivative parameters. Depending on the need of the control system, any of these parameters can be zero, such that the integral parameter does not affect the control law (proportional-derivative PD), nor the derivative parameter (proportional P). Each of these parameters influences the behavior of the system.

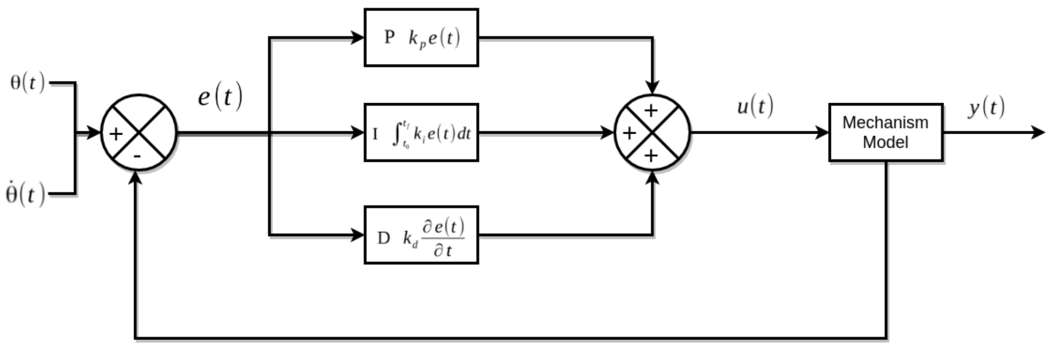


FIGURE C.1: A block diagram of a PID controller in a feedback loop.

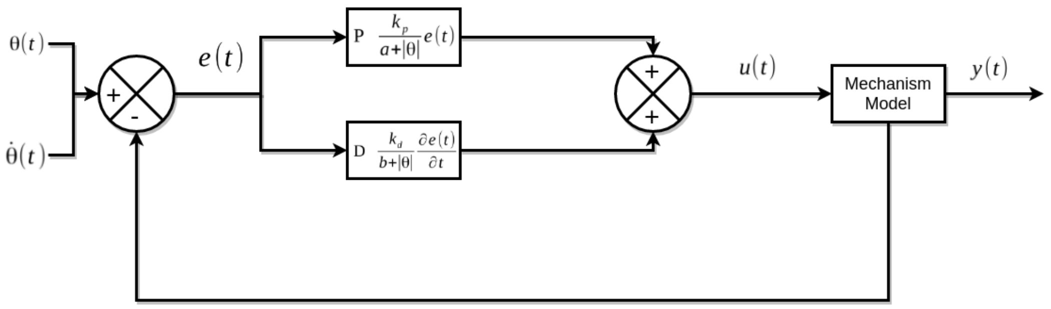
Figure C.1 shows a graphical representation of an standard PID control loop. As shown, the PID controllers start from an initial condition and it is continuously calculating an error value, computed as the difference between the desired value and a sensed variable. The controller attempts to minimize the error over time by enforcing asymptotic convergence of the sensed variable to the desired value. Let us set $\mathbf{q}$ as the sensed variable, in our cases of study, it can be either the position and orientation of the end-effector or the joint positions value. Then, the control input is given by:

$$\mathbf{u}(t) = \mathbf{k}_p \mathbf{e}(t) + \mathbf{k}_d \dot{\mathbf{e}}(t) + \mathbf{k}_i \int_{t_0}^{t} \mathbf{e}(t) \partial t \tag{C.1}$$

where $\mathbf{k}_p$, $\mathbf{k}_d$ and $\mathbf{k}_i$ are positive diagonal matrices related to the position (proportional), velocity (differential of position), and integral terms of error, respectively. $\mathbf{e}(t) = \mathbf{q}(t) - \mathbf{q}^D(t) \in \mathbb{R}^n$ and $\dot{\mathbf{e}}(t) = \dot{\mathbf{q}}(t) - \dot{\mathbf{q}}^D(t) \in \mathbb{R}^n$ are the joint translation and velocity errors, respectively. $\mathbf{q}(t)$ and $\dot{\mathbf{q}}(t)$ are the translation and velocity coordinates (if $\mathbf{q}$ corresponds to the data of the end-effectors, orientation is also included in both vectors) of the mechanism at an instant t. Likewise, $\mathbf{q}^D(t)$ and $\dot{\mathbf{q}}^D(t)$ are the desired translation and velocity coordinates imposed to the manipulator by the desired trajectory.

C.2 Adaptive proportional-derivative controller

Commonly, when applied in industry, PID controller gains are usually scheduled according to the process value and the magnitude of the error in order to overcome the non-linearity of the plant. The disadvantage of this method is that the tuning of the optimal set of gains based on empirical tests, that are not easy and time-consuming. Moreover, the conventional PID has a limited robustness to uncertainties in the model parameters, i.e., noise, unmodeled dynamics or uncertainty in parameters of the system. Thus, an adaptive PID as introduced by [55] is commonly used. The main idea is to create a continuous dynamic non-linear control system instead of gain-scheduling by creating a non-linear gain function, to achieve a better tracking and noise rejection.



$\theta(t)$
$\dot{\theta}(t)$
+
-
$e(t)$
P $\frac{k_p}{a+|\theta|}e(t)$
D $\frac{k_d}{b+|\theta|}\frac{\partial e(t)}{\partial t}$
+
+
$u(t)$
Mechanism Model
$y(t)$

Figure C.2: A block diagram of an adaptive PID controller in a feedback loop.

Fig. C.2, presents the control loop of an adaptive PD controller. An adaptive PID control can be described as a classical PID controller that uses non-linear gain. The adaptive control computes its control parameters inversely proportionally to the error of the sensed variable. Similar to the previous controller, the adaptive controller is given by:

$$\mathbf{u}(t) = \tilde{\mathbf{k}}_p(\mathbf{e})\mathbf{e}(t) + \tilde{\mathbf{k}}_d(\mathbf{e})\dot{\mathbf{e}}(t) \tag{C.2}$$

where

$$\tilde{\mathbf{k}}_p(\mathbf{e}) = \frac{\mathbf{k}_p}{(a + |\mathbf{e}|)} \qquad \tilde{\mathbf{k}}_d(\mathbf{e}) = \frac{\mathbf{k}_d}{(b + |\mathbf{e}|)}$$

$\mathbf{k}_p$ and $\mathbf{k}_d$ are constant positive diagonal matrices related to the position and velocity, respectively. a and b are positive constants. $\mathbf{e}(t) = \mathbf{q}(t) - \mathbf{q}^D(t) \in \mathbb{R}^n$ and $\dot{\mathbf{e}}(t) = \dot{\mathbf{q}}(t) - \dot{\mathbf{q}}^D(t) \in \mathbb{R}^n$ are the joint translation and velocity errors, respectively. $\mathbf{q}(t)$ and $\dot{\mathbf{q}}(t)$ are the translation and velocity coordinates (including orientation in both cases when feedback of data of the end-effector is used) of the mechanism at an instant t. Likewise, $\mathbf{q}^D(t)$ and $\dot{\mathbf{q}}^D(t)$ are the desired translation and velocity coordinates imposed to the manipulator by the desired trajectory.

Bibliography

[1] Martin Hägele. World robotics 2010: service robots; statistics, market analysis, forecasts, case studies.

[2] Adrian Bejenaru. Abb irb 2600. `https://grabcad.com/library/abb-irb-2600-1`, 2016. [Online; upload 2016-03-19].

[3] Mohamed Sami. Delta robot. `https://grabcad.com/library/delta-robot--2`, 2012. [Online; upload 2012-12-04].

[4] Kalyanmoy Deb and Santosh Tiwari. Omni-optimizer: A generic evolutionary algorithm for single and multi-objective optimization. *European Journal of Operational Research*, 185(3):1062–1087, 2008.

[5] S Ivvan Valdez, Arturo Hernández, and Salvador Botello. A boltzmann based estimation of distribution algorithm. *Information Sciences*, 236:126–137, 2013.

[6] R Clavel. Delta, a fast robot with parallel geometry. In *Proc. 18th Int. Symp. on Industrial Robots, Lausanne, 1988*, pages 91–100, 1988.

[7] CRISTHIAN Riaño, CESAR Peña, and ALDO Pardo. Approach in the optimal development of parallel robot for educational applications. In *Proceedings of the WSEAS international conference on Recent Advances in Intelligent Control, Modelling and Simulation (ICMS)*, volume 145, 2014.

[8] Virendra Kumar, Soumen Sen, Shibendu S Roy, Chandan Har, and SN Shome. Design optimization of serial link redundant manipulator: An approach using global performance metric. *Procedia Technology*, 14:43–50, 2014.

[9] Surabhi Patel and Tarek Sobh. Goal directed design of serial robotic manipulators. In *American Society for Engineering Education (ASEE Zone 1), 2014 Zone 1 Conference of the*, pages 1–6. IEEE, 2014.

[10] Sarosh Patel and Tarek Sobh. Task based synthesis of serial manipulators. *Journal of advanced research*, 6(3):479–492, 2015.

[11] YJ Lou, GF Liu, and ZX Li. A general approach for optimal design of parallel manipulators. *IEEE Transactions on Automation science and engineering*, 2005.

[12] A Omran, M Bayoumi, A Kassem, and G El-Bayoumi. Optimal forward kinematics modeling of stewart manipulator using genetic algorithms. *Jordan Journal of Mechanical and Industrial Engineering*, 3(4):280–293, 2009.

[13] Yunjiang Lou, Guanfeng Liu, and Zexiang Li. Randomized optimal design of parallel manipulators. *IEEE transactions on automation science and engineering*, 5(2):223–233, 2008.

[14] Yunjiang Lou, Yongsheng Zhang, Ruining Huang, Xin Chen, and Zexiang Li. Optimization algorithms for kinematically optimal design of parallel manipulators. *IEEE Transactions on Automation Science and Engineering*, 11(2):574–584, 2014.

[15] Akansel Cosgun, Harvey Lipkin, and Henrik I Christensen. Price-based optimization of serial robot manipulators under payload and workspace constraints.

[16] Xin-Jun Liu and Jinsong Wang. A new methodology for optimal kinematic design of parallel mechanisms. *Mechanism and Machine Theory*, 42(9):1210–1224, 2007.

[17] Stefano Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, 1997.

[18] T Ravichandran, GR Heppler, and DWL Wang. Task-based optimal manipulator/controller design using evolutionary algorithms. Technical report, University of Waterloo, 2004.

[19] Youshen Xia and Jun Wang. A dual neural network for kinematic control of redundant robot manipulators. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 31(1):147–154, 2001.

[20] Gilberto Reynoso-Meza, Javier Sanchis, Xavier Blasco, and Miguel Martínez. Algoritmos evolutivos y su empleo en el ajuste de controladores del tipo pid: Estado actual y perspectivas. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 10(3):251–268, 2013.

[21] Rogério Rodrigues dos Santos, Valder Steffen, and Sezimária de Fátima Pereira Saramago. Optimal task placement of a serial robot manipulator for manipulability and mechanical power optimization. 2010.

[22] M Pellicciari, G Berselli, F Leali, and A Vergnano. A method for reducing the energy consumption of pick-and-place industrial robots. *Mechatronics*, 23(3):326–334, 2013.

[23] Thambirajah Ravichandran, David Wang, and Glenn Heppler. Simultaneous plant-controller design optimization of a two-link planar manipulator. *Mechatronics*, 16 (3):233–242, 2006.

[24] Helon Vicente Hultmann Ayala and Leandro dos Santos Coelho. Tuning of pid controller based on a multiobjective genetic algorithm applied to a robotic manipulator. *Expert Systems with Applications*, 39(10):8968–8974, 2012.

[25] EL Badreddine, Ajmi Houidi, Zouhaier Affi, and Lotfi Romdhane. Application of multi-objective genetic algorithms to the mechatronic design of a four bar system with continuous and discrete variables. *Mechanism and Machine Theory*, 61:68–83, 2013.

[26] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control.* Springer Science & Business Media, 2010.

[27] MW Spong and M Vidyasagar. *Robot dynamics and control.* John Wiley & Sons, 2008.

[28] John J Craig. *Introduction to robotics: mechanics and control*, volume 3. Pearson Prentice Hall Upper Saddle River, 2005.

[29] Charles A Klein and Bruce E Blaho. Dexterity measures for the design and control of kinematically redundant manipulators. *The International Journal of Robotics Research*, 6(2):72–83, 1987.

[30] M López, E Castillo, G García, and A Bashir. Delta robot: inverse, direct, and intermediate jacobians. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 220(1):103–109, 2006.

[31] John Stephen Berry. *Dictionary of Mathematics.* Taylor & Francis, 1999.

[32] Krzysztof J Kaliński and Michał Mazur. Optimal control at energy performance index of the mobile robots following dynamically created trajectories. *Mechatronics*, 2016.

[33] Pedro Larranaga and Jose A Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer Science & Business Media, 2002.

[34] Hans-Paul Paul Schwefel. *Evolution and optimum seeking: the sixth generation.* John Wiley & Sons, Inc., 1993.

[35] Lawrence Davis. Handbook of genetic algorithms. 1991.

[36] A Klanac and J Jelovica. A concept of omni-optimization for ship structural design. *Advancements in Marine Structures, Proceedings of MARSTRUCT*, pages 473–481, 2007.

[37] Nikolaus Hansen, André SP Niederberger, Lino Guzzella, and Petros Koumoutsakos. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. *IEEE Transactions on Evolutionary Computation*, 13(1):180–197, 2009.

[38] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001.

[39] Nidamarthi Srinivas and Kalyanmoy Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[40] Z Konfrst. Parallel evolutionary algorithms: advances. In *Automation Congress, 2004. Proceedings. World*, volume 17, pages 429–434. IEEE, 2004.

[41] Erick Cantú-Paz. Designing scalable multi-population parallel genetic algorithms. 1998.

[42] Theodore C Belding. The distributed genetic algorithm revisited. *arXiv preprint adap-org/9504007*, 1995.

[43] V Scott Gordon and Darrell Whitley. Serial and parallel genetic algorithms as function optimizers. In *ICGA*, pages 177–183, 1993.

[44] Shumeet Baluja. Structure and performance of fine-grain parallelism in genetic search. In *ICGA*, pages 155–162, 1993.

[45] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7:15, 2008.

[46] Charles Miller Grinstead and James Laurie Snell. *Introduction to probability*. American Mathematical Soc, 2012.

[47] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[48] Richa Sharma, KPS Rana, and Vineet Kumar. Comparative study of controller optimization techniques for a robotic manipulator. In *Proceedings of the Third International Conference on Soft Computing for Problem Solving*, pages 379–393. Springer, 2014.

[49] Roger Fletcher and Michael JD Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163–168, 1963.

[50] David F Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656, 1970.

[51] Enrique Alba. Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters*, 82(1):7–13, 2002.

[52] Themes & Co. 2016 ieee la-cci (latin american conference on computational intelligence). http://la-cci.org/, 2016. [Online; upload 05-11-16].

[53] Vigen Arakelian, Jean-Paul Le Baron, and Pauline Mottu. Torque minimisation of the 2-dof serial manipulators based on minimum energy consideration and optimum mass redistribution. *Mechatronics*, 21(1):310–314, 2011.

[54] Hui-Hung Lin, Chih-Chin Wen, Shi-Wei Lin, Yuan-Hung Tai, and Chao-Shu Liu. Robust control for a delta robot. In *SICE Annual Conference (SICE), 2012 Proceedings of*, pages 880–885. IEEE, 2012.

[55] Jingqing Han. From pid to active disturbance rejection control. *IEEE transactions on Industrial Electronics*, 56(3):900–906, 2009.