ITALIAN INSTITUTE OF TECHNOLOGY, AND
UNIVERSITY OF GENOVA

Mechanism and Behaviour Co-optimisation of High Performance Mobile Robots

by

Antonios Emmanouil Gkikakis

Thesis submitted for the degree of *Doctor of Philosophy* (33° cycle)

April 2021

Department of Informatics, Bioengineering, Robotics and Systems Engineering
PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

Professor Roy Featherstone	Supervisor
Professor Darwin Caldwell	Supervisor
Professor Giorgio Cannata	Head of the PhD program

Thesis Jury:

Professor Sehoon Ha, *Georgia Institute of Technology*	External examiner
Professor Cesare Stefanini, *Khalifa University*	External examiner

Table of contents

List of figures

List of tables

Acknowledgements

This thesis becomes reality with the help and the support of many individuals. I would like to extend my sincere thanks to all of them.

Firstly, I am highly indebted to my supervisor Professor Roy Featherstone, for guiding me through my PhD, and for spending hours to teach me about dynamics, control, calculus, geometry, algebra, software engineering, nuclear physics, and all those fun stuff. I would like to thank my teammate and friend Roodra Pratap Singh Bajwa for the fun and yet laborious workdays, weekends and holidays that we spent together in this journey to obtain our PhDs. I also wish to acknowledge the help provided by my teammates Federico Allione and Juan David Gamba Camacho. Special thanks to the external reviewers of my thesis, Professor Sehoon Ha, and Professor Cesare Stefanini for their insightful and critical reviews.

I am extremely grateful to my mother and father, Panagiota Gkikaki Triantafullou and Marios Gkikakis, who have always been by my side, and have supported me throughout my life. It wouldn't have been possible for me to reach that far without them.

I would like to thank my dearest friends: Konstantinos Kapasakalis, Bill and Eleftherios Tzallas for continuously motivating me to surpass my limits both mentally and physically. I would also like to thank Antonios Mastropetros, Stefanos Ioakim Kalapothakis, Jim Makrinikolas, Alexandros Vergopoulos and Anastasia Rokka for being awesome friends, and for believing in me.

I would like to express my special gratitude to the people that I spent the most time with here in Genova (which happened to be the most fun time too), Joao Bimbo, Mehrdad Tavassoli, Giulia Mazzon, Olmo Alfonso Moreno Franco, and Dimitrios Kanoulas for his valuable advice and guidance. Finally, many thanks to the stuff and technicians of IIT, and UniGe for providing me with technical support throughout these years.

Abstract

Mobile robots do not display the level of physical performance one would expect, given the specifications of their hardware. This research is based on the idea that their poor performance is at least partly due to their design, and proposes an optimisation approach for the design of high-performance mobile robots. The aim is to facilitate the design process, and produce versatile and robust robots that can exploit the maximum potential of today's technology. This can be achieved by a systematic optimisation study that is based on careful modelling of the robot's dynamics and its limitations, and takes into consideration the performance requirements that the robot is designed to meet. The approach is divided into two parts: (1) an optimisation framework, and (2) an optimisation methodology. In the framework, designs that can perform a large set of tasks are sought, by simultaneously optimising the design and the behaviours to perform them. The optimisation methodology consists of several stages, where various techniques are used for determining the design's most important parameters, and for maximising the chances of finding the best possible design based on the designer's evaluation criteria.

The effectiveness of the optimisation approach is proved via a specific case-study of a high-performance balancing and hopping monopedal robot. The outcome is a robot design and a set of optimal behaviours that can meet several performance requirements of conflicting nature, by pushing the hardware to its limits in a safe way. The findings of this research demonstrate the importance of using realistic models, and taking into consideration the tasks that the robot is meant to perform in the design process.

Part I

Robot Design Optimisation Approach

Chapter 1

Introduction

Recent technological innovations and an increasing demand for automation have lead to a rapid evolution of the field of robotics; robots find applications in industrial environments [54], surgery rooms [72], space exploration [36] and various other scenarios. Robot designs span from industrial grippers [8], mobile rovers and legged robots [89], soft robots [90] and more. Each application might require a different design, which can be task specific. For example, industrial robotic arms could be used for packing, selecting or assembling products. These robots have usually a fixed base, and are generally very precise, durable and display highly repeatable behaviours. However, they are mostly confined in industrial environments, due to their lack of mobility.

On the other hand, mobile robots, and especially legged ones, offer high mobility and traversability, but are more challenging to design due to the dynamic nature of locomotion, and are difficult to control because they can fall and damage themselves or their surroundings. This leads to a higher production cost, and increased mechanical, electronic and software complexity. One can easily notice that commercially available robotic arms are widely used in industry and elsewhere; in contrast to commercially available legged robots, which have only recently entered the market.

Motivated by the numerous applications and benefits legged robots can offer, and the lack of complete and systematic approaches for building such robots, an optimisation approach to facilitate the design of high-performance legged robots is proposed in this thesis. This work contributes to the field of robot design optimisation of legged robots by presenting an in-depth analysis on the design of a monopedal robot which is simple because it has only one leg, but is governed by complex dynamics and is difficult to control due to its instability. The results of this work provide a deeper insight in the challenges faced during the design process of legged robots, and can be used for the design of more complicated and potentially more useful machines, such as bipedal robots.

1.1 Topic

The topic of this thesis is the design of versatile high-performance mobile robots with the use of realistic models and numerical optimisation techniques. The study treats together the optimisation

of the design and its behaviours for achieving a set of performance requirements. The design and behaviour co-optimisation is complemented by a meticulous optimisation approach that takes into account the robot's most critical components and limitations, the performance requirements, the amount of available resources, and manufacturability criteria for the purpose of obtaining robot designs that meet their design expectations, with fewer design iterations. This approach can drastically reduce the production cost of legged robots. As a specific example, the proposed optimisation approach is applied to the design of a high-performance balancing and hopping one-legged robot, named Skippy.

Legged robots have been a topic of interest since the early 1980s, when Raibert [83] introduced a 2 D and a 3 D balancing and hopping machine. However, several decades had to pass until significant progress was observed in the field of legged locomotion. Only recently, a miniature robot named Salto displayed precise hopping and balancing skills in the work of Yim et al. [111] surpassing the work of Raibert. Furthermore, advancements in robot design optimisation were presented in the work of Ha et al. [46], where numerical optimisation techniques were used for the design and behaviour co-optimisation of various types of robots. On the same line, the research presented in this thesis focuses on maximising the physical ability of mobile robots with the use of optimisation techniques. One of the key differences between the presented study and current literature is that realistic models of critical components of the robot and their limitations are taken into consideration for achieving multiple behaviours of conflicting nature. The study takes into account the design and the tasks that the robot is designed to perform, so that the final design can meet a large set of performance requirements that are close to the robot's maximum physical potential. To achieve this, realistic models of the robot's hardware and its behaviours must be built, and a thorough optimisation study for investigating the robot's maximum potential must be performed.

1.2 Aim

The aim of this thesis is to demonstrate that legged robots are under-performing due to poor design decisions, by introducing a new way for designing them. To do this, an optimisation approach is proposed to facilitate the design process of high-performance legged robots. This approach aims to investigate how to maximise the physical ability of legged robots to safely reach their maximum physical potential, and reduce the gap between simulation and reality. To achieve this, careful modelling of the robot's dynamics is necessary, which can lead to robot designs that meet their design expectations in fewer design cycles.

Nowadays, technology offers powerful, light and precise actuators, sophisticated sensors, and a variety of materials with amazing mechanical properties, such as fibre composites or very strong and light alloys. The available technology exists to allow robots to be much faster and more robust than they currently are. So why are legged robots not widely used yet, and cannot reach the level of performance one would expect given the specifications of their individual parts? Furthermore, what would be a scientific approach to design robots that meet design expectations, without the need of

several costly design iterations? Moreover, how can we create versatile robots that can be used in a variety of applications and different tasks? Following these research questions, the main hypothesis of the thesis is formed, which is that the poor performance of legged robots is a result of poor design decisions, that were not a result of a proper scientific analysis of its mechanism and its performance objectives.

State-of-the-art research focuses on robot designs that are optimised to perform a relatively small set of tasks, and as a result the effect of different behaviours on the design of the robot is not properly explored. In addition, several design studies neglect to model behaviours and limitations of important components of the robot, resulting in designs that do not meet their performance requirements in reality. Finally, a robot's design has an inherent relationship with the tasks it is designed to perform. This means that different designs might perform better or worse in various tasks. For this reason the design and the behaviours of the robot must be treated together and co-evolve for obtaining capable robots; something that is not always the case in the robot-design literature.

1.3 Contribution

Robots are intelligent machines designed to increase the human standard of living and make our lives safer. The main contribution of this thesis is a robot-design optimisation approach to facilitate the design process of legged robots for making them more robust, reliable, and capable for the tasks that they are designed to perform. Thus, a new way of designing mobile robots may be arrived at. In addition, the findings of the examined case-study, which focuses on the design of a one-legged robot, can provide insights on the fundamental principles of legged locomotion, and aid or be directly applied to the design of more complicated legged robots such as bipeds, that can find immediate applications in numerous real-life scenarios.

Robot design is the discipline where intelligent machines are designed for the purpose of serving and assisting humans. In the design process multiple mechanical and electrical parts, each with its own complexity, capabilities and limitations are combined to create a robot. However, the complexity of all these components, together with the robot's intended use, make robot design a challenging task.

To overcome this issue, a more systematic approach must be followed. A robot's most critical parts must be accurately modelled, so the design can be tested in realistic virtual environments. This procedure can be much faster and cheaper than building and testing multiple real prototypes, and can even result in unprecedented performance, for the simple reason that many more robots can be designed and tested in simulation, rather than would have been possible in reality.

To address the design problem, an optimisation approach is proposed, which is divided into two parts: (1) an optimisation framework, and (2) an optimisation methodology. The optimisation framework is designed to exploit the inextricable relationship between a design and its behaviours by splitting the process into two interconnected optimisation layers. In the first layer, robot designs that can achieve a set of performance requirements are sought, and in the second layer the design's best

behaviours to achieve these requirements are sought. The result is a set of robot designs that can meet the required performance objectives, and the optimal behaviours to achieve them.

This is an example of mechanism and behaviour co-optimisation, which means a joint or mutual optimisation of both the behaviour and the mechanism of the robot. Initially, only the performance objectives are given, along with those aspects of the design that have already been decided. The objective is then to find both the physical parameters of the best mechanisms and the behaviour parameters of their best behaviours. Observe that the term 'behaviour' is used to describe what exactly the robot does, and the term 'performance objective' to describe the required outcome of that behaviour. A behaviour is then awarded a score according to how well it meets its objective; and the lower layer optimiser seeks the behaviour with the best score. Furthermore, the proposed approach optimises the mechanism of a robot design; however, it is not limited to only that. The definition of a mechanism excludes parts such as actuators, power supplies and sensors; but these parts and their limitations are taken into account in the proposed approach.

Together with the optimisation framework, a methodology for performing the experiments is proposed. The methodology aims: (1) to increase the chances of finding the best possible design (the search space is substantially large in most cases) for the desired tasks, (2) to help gain a deeper understanding of the examined system, so that appropriate design decisions can be made, (3) to obtain a final design that is robust, and can achieve the desired performance, even in the presence of manufacturing and modelling errors.

A limitation of this approach is that significant effort is required for obtaining and implementing realistic models of the robot's hardware. For example, to model the behaviour of some components (e.g., springs) specialised equipment might be required (e.g., tensile measurement machines) in order to estimate model parameters. In addition, a deep understanding of a robot's hardware is required, and development of sophisticated software for the robot's model. Furthermore, accurate models are computationally expensive, which might result in experiments which last long periods of time. Nevertheless, an experienced designer in an appropriate workplace can overcome all of these limitations.

The effectiveness of the proposed optimisation approach is shown via a case study on the design of a high-performance robot. The study focuses on legged robots, and more specifically on the design of a versatile and agile one-legged robot that is capable of achieving multiple athletic feats. The final result is an optimal design capable of meeting a large number of performance objectives. The behaviours of this design are presented in simulation, and demonstrate that the proposed approach can result in robot designs that can display a plethora of impressive athletic behaviours, which come close to the maximum potential of their individual components in a safe way.

1.4 Outline

This thesis is divided into two parts. The first part presents a general idea for robot design optimisation, and the second part presents a specific example of the general idea via a case study on a one-legged

hopping and balancing robot. The rest of this thesis is organised as follows.

Part I: Robot Design Optimisation Approach. This part provides a broad introduction to the field of mathematical optimisation, robot design optimisation, and introduces the proposed design philosophy.

Chapter 2: Background: Mathematical Optimisation. Introduction to mathematical optimisation and important concepts. The chapter presents various optimisation problem types and approaches so that the different problems in robot design can be classified and addressed.

Chapter 3: Related Works: Robot Design Optimisation. Related works in the field of robot design optimisation are presented and critically evaluated. A comparison with the proposed approach is presented and a discussion on the gaps this study helps to fill.

Chapter 4: Optimisation Framework and Methodology. The main contribution of this thesis is presented, which is a robot design optimisation approach, which consists of a framework and a methodology.

Part II: Case study: Skippy, a hopping Monopedal Robot. In this part, the proposed design optimisation philosophy is applied to the design of a high-performance monopedal robot.

Chapter 5: Building a Model: Hardware and Behaviour. This chapter presents the hardware and behaviour models of the Skippy robot, which is a one-legged balancing and hopping machine.

Chapter 6: A Versatile Hopper. The proposed optimisation approach is applied to the design optimisation of Skippy. The objective is to obtain a single design capable of achieving a large set of demanding performance objectives, including a 3 m vertical hop and a 2 m triple somersault. The final result is an optimal design and its optimal behaviours, which are presented and analysed.

Chapter 7: Discussion

Chapter 8: Conclusion

Chapter 2

Background: Mathematical Optimisation

Optimisation is the act of making something better. An example of optimisation is trial and error, which is the oldest and most commonly used method for improving something. However, trial and error is mainly based on arbitrary decisions, and can be ineffective when one is facing complicated problems. With the advance of science and engineering, the former 'art' of optimisation, which was based solely on one's experience and imagination, started to acquire a more systematic and quantitative form, and was coined with the term *mathematical optimisation* or *mathematical programming*. As the name suggests, mathematical optimisation is concerned with finding the best value of a mathematical formula. This formula, which is also called the *objective function*, could be a single function or even a complicated model of a real system that consists of many interconnected formulas and conditions. The *output value* of the objective function is the value we wish to optimise, and this can be achieved by finding the *input value* to do so. The simplest optimisation problem is to find a formula's maximum or minimum point, such as minimising the manufacturing cost of a product.

Optimisation formulae were proposed as early as the 1630s by Pierre De Fermat with *Adequality* and the *Euler-Lagrange equation* for calculating the maximum or minimum of a mathematical function in the 1750s. Iterative methods also appeared in 1600s with *Newton's Optimisation Method* for finding the roots of differentiable functions and the *Gauss Method* (1800s) for solving the least squares problem, which is used for minimising the error in data fitting applications, such as determining the orbit of celestial bodies.

Several advancements were made in mathematical optimisation during the 20th century, where the field started to become more formalised. The invention of methods such as the *Simplex Algorithm* by Dantzig (1947) to solve linear optimisation problems; the *duality principle* discovered by John von Neumann, also in 1947, which provides an upper or a lower limit to the value of an optimisation problem; and the *Karush–Kuhn–Tucker Conditions* for optimality (1951) further popularised the field and its applications. The introduction of computers and the large computational power that came with them facilitated the application of mathematical optimisation and allowed its widespread use.

Today, optimisation is an essential part of numerous science and engineering fields such as: astronomy, geodesy, medicine, manufacturing, transportation, finance, economics, artificial intelligence, robotics and more.

2.1 Optimisation Problems

Optimisation problems can be categorised into a plethora of different fields and sub-fields based on a set of characteristics. These characteristics include the nature of the problem; for example, is it deterministic or non-deterministic (i.e., is there noise in the system), is it differentiable, is it linear or non-linear and so on.

The aim of this chapter is to introduce the reader to the field of mathematical optimisation and its most important problems, methods, applications and concepts. The information presented in this chapter is essential for understanding why optimisation of robot design is a difficult problem to solve, and how it can be approached. Mathematical optimisation is a vast field, so elaborate details about concepts and proofs are kept to minimum to preserve the compactness of this thesis.

In this thesis only the main optimisation fields and sub-fields will be discussed with a focus on the most relevant ones for robot design. Classifying an optimisation problem with other similar problems is an important step to understand how challenging would be to solve it, and consequently to estimate the resources to do so. Furthermore, many methods are tailored for solving specific types of optimisation problems, and thus the selection of the most appropriate algorithm can result in significantly better and faster results. For this reason, in this chapter various problem types are presented with some example applications that demonstrate why solving each problem is important. This section presents the basic concepts and mathematical definitions of optimisation problems as well as a short discussion on the different types that exist. The next section will deal with the various methods for solving optimisation problems and lay out the basic concepts for selecting the most appropriate method for a given problem.

Before the miscellaneous optimisation sub-fields are discussed, the basic concepts of optimisation and a formal mathematical definition must be presented. A general mathematical formalisation of an optimisation problem can be the following:

$$\begin{aligned} \min_{x} \quad & f_1(x), f_2(x), ..., f_i(x) && i \in I \\ \text{s.t.} \quad & g_j(x) \leq G_j, && j \in J \\ & h_k(x) = H_k, && k \in K \\ & x_L \leq x \leq x_U, && x_L, x_U \in \mathbb{R}^n. \end{aligned} \tag{2.1}$$

An optimisation problem can be described by the following.

1. x: are the input or independent variables. Depending on the problem $x \in \mathbb{R}^n$, or $x \in \mathbb{N}^n$, or may be a mixture of discrete and continuous variables.

2. $x_L \leq x \leq x_U$ is the set of all valid input values, which can be represented by the symbol Ω. x_L is a lower bound and x_U is an upper bound to the input variables.

3. f_i: are the objective functions with $f_i : \mathbb{R}^n \to \mathbb{R}$, which are to be minimised or maximised. I is the set of indices for objective functions.

4. g_j: are the inequality or non-binding constraint functions with $g_j : \mathbb{R}^n \to \mathbb{R}$. Every value of $g_j(x)$ must be less than or equal to the value of G_j. J is the set of indices for inequality constraints.

5. h_k: are the equality or binding constraint functions with $h_k : \mathbb{R}^n \to \mathbb{R}$. Every value of $h_k(x)$ must be equal to the value of H_k. K is the set of indices for equality constraints.

Feasible set or F is the set of all solutions in Ω that satisfy all the constraints in Equation 2.1. A problem is unfeasible (there is no solution to it) when $F = \emptyset$. If K and J are empty (there are no constraints) then $F = \Omega$.

Optima are the solutions of interest in an optimisation problem. Optima can be found at critical points.

Critical point of a mathematical function is a point where the derivative does not exist or it is equal to zero. If the function is differentiable and the first derivative at that point is zero then it is called a *stationary point*. There are three different types of critical points.

1. *Minimum points*, which are also extremum points. In these type of problems the goal is to find the minimum value of the objective function (e.g., minimise the cost of production).

2. *Maximum points*, which are also extremum points. Problems where the maximum value is sought (e.g., maximise the performance of a robot).

3. *Saddle points* or *equilibrium points*. These points are not extrema and are solutions of interest in equilibrium optimisation problems. These type of problems examine the interaction or collective behaviours of multiple entities with each entity seeking to minimise or maximise a set of objective functions. Equilibrium optimisation problems are studied in the field of *Game Theory*, which finds applications in economics and cooperative games [18]. This thesis does not address equilibrium optimisation problems, and hence for the rest of this thesis the term optimisation problem will always refer to a maximisation or minimisation problem.

Existence of optima—one of the many challenges in mathematical optimisation is that in most cases there is no prior knowledge regarding the existence of optimal solutions. This means that an optimal solution may not exist at all (see left plot in Figure 2.1), and searching for something that does not exist results in wasted resources.

If certain conditions are met the *Extreme Value Theorem (Weierstrass Theorem)* can guarantee the existence of optima. Specifically if the objective function f is continuous and the set of feasible

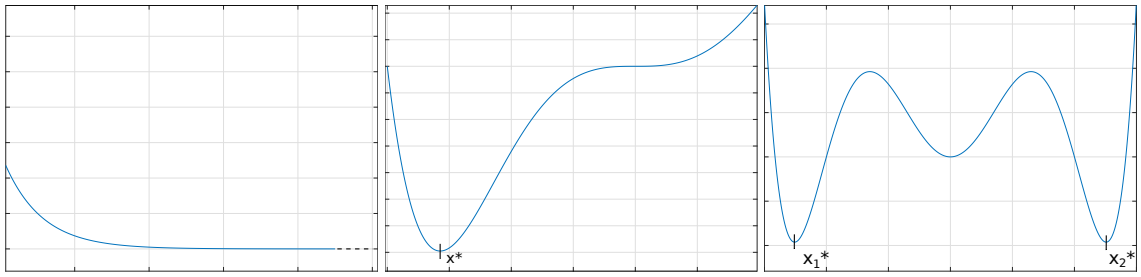

Figure 2.1: Existence of optima. Left: no global optimum exists (function e^{-x} goes to zero as x goes to ∞); centre: a single optimum exists at x^*; right: two global optima exist at x_1^* and x_2^*.

values Ω is compact (bounded and closed) then it has at least one maximum and minimum value. However, in complex problems where little information is known about the objective function and its properties it is difficult to know in advance if any optima exist. Figure 2.1 demonstrates the possible cases about the existence of an optimum, which are:

1. it exists and it is unique,
2. it exists but it is not unique,
3. it does not exist.

The most ideal scenario would be the first one. Several problems exist though that have many optimal solutions.

Unimodal—a function $f : \mathbb{R} \to \mathbb{R}$ is unimodal when for a value $m \in \mathbb{R}$ is monotonically increasing for $x \leq m$ and monotonically decreasing for $x \geq m$ $\forall x \in \mathbb{R}$, or alternatively:

$$m \in \mathbb{R} \text{ s.t. } f(x) \searrow \forall x \leq m \text{ and } f(x) \nearrow \forall x \geq m. \tag{2.2}$$

In this case $f(m)$ is an extremum point of f that has the smallest value over its domain and is a minimum and a *global optimum* [12]. All the definitions and examples in this chapter will be given for the minimum extremum, unless stated otherwise; however, the same reasoning holds for maximum extremum. If x is a vector and not a scalar the function is called *Quasiconvex*.

Multimodal—a function $f : \mathbb{R} \to \mathbb{R}$ is multimodal when $\nexists\, m \in \mathbb{R}$ that can satisfy Equation 2.2 for the entire domain of f. Such functions can be locally unimodal in a defined interval $[a,b]$ for $a,b \in \mathbb{R}$. In this case, the local value m, which is a local extremum is also called *local optimum*. A multimodal function may have several local optima out of which one or more can have the single best value. The extremum points with the best value are called *global optima*. If x is a vector and not a scalar the function is called *Non-Quasiconvex*.

As shown in the definition of an optimisation problem (Equation 2.1) depending on the examined problem there can be one, many or no objective functions, and for the first two cases the concept of a global optimum varies.

No objective function—in this case the objective is to find the *Feasible Set* of solutions F, if there exist any.

Single-Objective Function—when there is only one objective to optimise. For this type of problems the optima can be defined as following.

- **Local optimum** is a solution that cannot be improved by values in its close proximity. Alternatively for an objective function $f : \Omega \to \mathbb{R}$ a point $x^* \in \Omega$ is a local minimum:

$$\text{if } \exists\ \delta > 0 \text{ s.t. } f(x - x^*) \geq f(x^*)\ \forall\ x \text{ with } \|x - x^*\| < \delta. \tag{2.3}$$

  Multi-modal objective functions can have several local optima. A local optimum that has the best value of all the local optima is also called a *global optimum*.

- **Global optimum** is a solution $x^* \in \Omega$ that has the best value out of all feasible solutions or alternatively:

$$\forall x \quad \in \quad \Omega \quad f(x^*) \leq f(x). \tag{2.4}$$

A local optimum in a function with a unique optimum is also a global optimum.

Multi-objective functions—in this scenario obtaining a single optimal solution is not as straightforward as in the single-objective case. In an ideal case an optimal solution can be a solution $x^* \in \Omega$ that outperforms every other solution in each of the objectives.However, this is rarely the case. The reason is that in many problems the objective functions are correlated and have a relationship of conflicting nature, which creates *trade-offs*. This means that improving one objective diminishes another one (or even more) so the problem of finding a single optimal solution becomes more complicated. As a result, the optimal solution depends on the *decision maker's* (usually a human) perspective and is usually the most acceptable solution based on the given circumstances. Examples of decision making are discussed in the case study presented in Part II. The set of all optimal solutions is called the *Pareto set* or the *Pareto front* [18].

Pareto front is the set of *non-dominated* solutions. A feasible solution is non-dominated when for a given solution no better value can be obtained unless the value of another objective is sacrificed. In a minimisation problem a solution x_1 *dominates* a solution x_2 if:

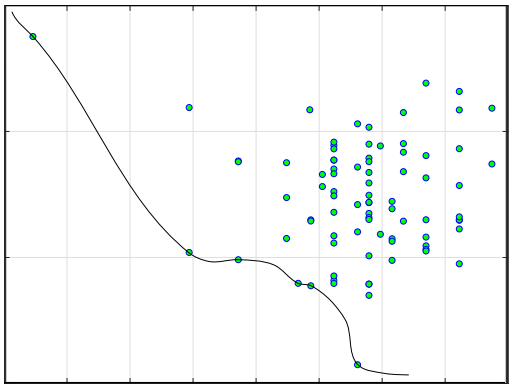

Figure 2.2: Example of feasible solutions in a bi-objective (has only two objectives) optimisation problem. The solutions that the black line passes through constitute the Pareto front of this minimisation problem.

$$\begin{aligned} &1. \quad f_i(x_1) \leq f_i(x_2)\ \forall i \in I, \text{ and} \\ &2. \quad \exists\ j \in I \text{ s.t. } f_j(x_1) < f_j(x_2). \end{aligned} \tag{2.5}$$

Nondominated solution is a solution x^* that is not dominated by any other feasible solution. These solutions constitute the Pareto front, which represents the solutions with the optimal trade-off between all the feasible solutions. An example of a bi-objective optimisation problem and its feasible solutions is presented in Figure 2.2. The Pareto front of optimal solutions includes all the points that the black line passes through.

Multi-objective problems usually involve multiple parameters (high dimensionality of input and output variables) and are inherently difficult to solve due to the fact that no efficient methods for solving them exist, which will be discussed in more detail in the next section. This thesis deals with this broad problem for the design optimisation of robots with the purpose of creating versatile robots that can achieve a variety of conflicting objectives.

Proof of Optimality—as it is explained in the next section, a variety of methods exist for solving optimisation problems. All of these methods may return a solution or a set of solutions for a given problem. The question that naturally arises is: given a point, can we prove that this point is an optimum (local or global)? The answer is yes, but only under certain conditions.

In the case of continuously differentiable, and constrained or unconstrained problems there exists a set of *necessary* conditions that a point must meet to be a stationary point, and are called the *First order* or *Karush Kuhn Tucker Optimality Conditions* (KKT). To identify if a stationary point is a maximum, a minimum or a saddle point the *Second Order Sufficient Conditions* must be met, which require the second-order derivatives of the objective function, and can only be applied to unconstrained problems. Proofs and details about optimality conditions can be found in Boyd and Vandenberghe [12].

The drawback of such methods is that they require smooth objective functions and information about their first and second derivatives, which is difficult to obtain in many problems. Furthermore, these methods do not provide any information about local or global optimality. The absence of *global optimality conditions* makes optimisation problems even tougher to solve due to the uncertainty behind the global optimality of discovered solutions.

2.1.1 Computational Complexity

Before presenting the problem types in more detail, a short discussion about *computational complexity* is essential for understanding why some type of problems are more difficult to solve, and in which categories the problem of robot design and behaviour co-optimisation falls into. This is a broad field in mathematics and computer science, so only the most basic concepts that will help evaluate the difficulty of the examined problem are presented. A more elaborate discussion on the field of computational complexity can be found in Arora and Barak [1].

Computational complexity theory provides us with the necessary tools to quantify a problem's difficulty. With it we can measure the difficulty of computational problems based on the amount of resources that are required for solving them on a computer. This is useful for the following reasons: (1) we can compare problems according to their difficulty, and (2) we can select the most efficient methods for solving them, a topic that is discussed in the next section.

Models of computation are mathematical models that describe the input to output relationship including how computation, memory and communication is organised. These models allow problem types to be quantified based on their complexity. Complexity depends on the size of the input and the instance of the problem (e.g., different values of the input variables). The basic and most important factors taken into account when analysing the complexity of any method is its memory and time requirements.

The most commonly used model of computation is the *Turing Machine* (TM). These theoretical and general computing machines are popular for their simplicity, and the fact that any algorithm can be represented by one of them [1]. The reason why the TM is shortly introduced is because it serves as the basis for defining the complexity of a given problem family. The complexity of a given problem is defined by the amount of discrete steps and memory an algorithm requires to execute on a TM. Because the TM and computational complexity theory is out of the scope of this thesis only informal definitions will be presented for the sake of argument.

Tractable Problems are problems that do not require many resources to be solved. Specifically, tractable problems are the ones that have a polynomial input to output relationship. This means that the required time to solve these problems increases in a polynomial way with the input size. Problems with very high resource requirements are called *intractable*. Example of such problems

have exponential input to output relationships and computation of an exact solution is sometimes impossible for large input sizes.

Reduction is the process of transforming a problem A to another problem B. This process is important because it shows that problem A is *at most as difficult as* problem B, and if there is a method for solving the latter then the former can also be solved with it.

Computational problems are classified into five categories with one of them being *optimisation problems* which is the category of interest in this thesis. However, most of complexity theory is defined in terms of *decision problems* mostly because they are easier to understand [1]. For this reason, before defining the complexity of an optimisation problem, some terms in decision problems must be defined so they can be extended to the former.

Decision Problems are the family of problems that their answer is simply a *Yes* or *No*. Any optimisation problem can be reduced to a decision problem [2]. Their complexity can be classified in the following categories:

- P (polynomial time) complexity;
- NP (non-deterministic polynomial time) complexity;
- NP-complete;
- NP-hard.

P decision complexity problems are generally tractable and can be solved efficiently because they have a polynomial input to output relationship. This means that they can be solved in a fixed and predetermined amount of steps that does not change dramatically with the number of inputs.

NP complexity problems on the other hand cannot be solved efficiently (in polynomial time) by any known algorithm, and hence are more difficult to solve than P problems. These type of problems cannot be solved fast but given a possible solution it can be efficiently verified in polynomial time.

NP-complete is a decision problem that is: (1) NP, and (2) all problems in NP can be *reduced* to any problem belonging to the NP-complete complexity class. This class is an indicator of a problem's intractability meaning the best known algorithms for solving it exactly require at least exponential time.

NP-hard type of problems are at least as hard as the hardest problems in NP (Papadimitriou and Steiglitz [77]). This class does not only include decision problems, but also optimisation problems.

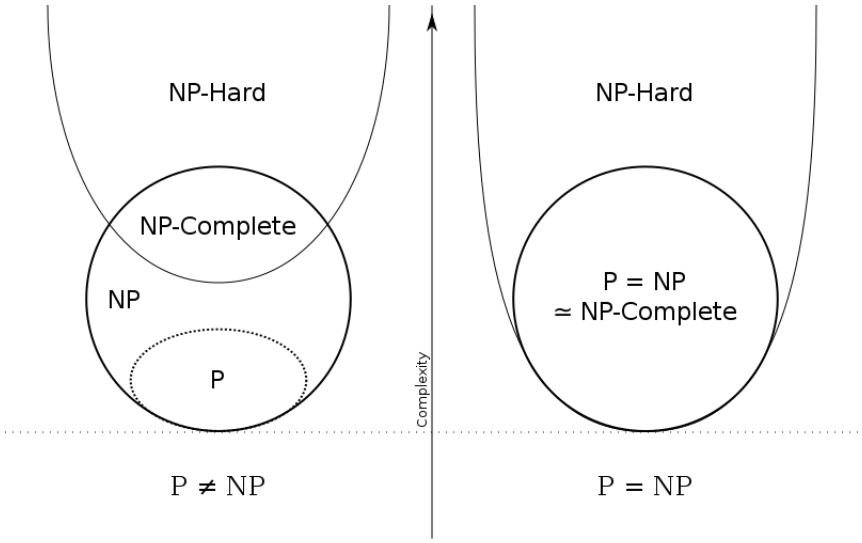

Figure 2.3: P Vs NP. The left side is valid if P≠NP and the right if P=NP.

P Vs NP—the aforementioned statements regarding the tractability of the presented problem families are based on the widely believed assumption that P≠NP. This unsolved problem is one of the most important problems in computer science and mathematics and argues if any problem in NP can be solved efficiently (in polynomial time). Unfortunately, the answer to this question is unknown at the time this thesis is written; however, it is widely believed that this assumption is not true [77], and in fact solving this *Millennium Prize Problem* [53] comes with a million dollar prize (there are seven of them and only one has been solved).

The consequences of this, are that when we face a general NP-hard optimisation problem we should abandon the idea of solving it exactly because the problem is computationally intractable for large input sizes. For this reason *approximation* methods or *heuristics* can be used that provide fast solutions that are *acceptable*. Approximation algorithms provide some mathematical guarantee of convergence for the returned solutions but heuristics on the contrary do not. This topic is discussed in the next chapter together with more details on the various techniques for solving optimisation problems.

2.1.2 Classification

An optimisation problem could be classified in any of the following categories

1. **Continuous** Vs **Discrete** Vs **Mixed Integer** Optimisation.
2. **Unconstrained** versus **Constrained** Optimisation.
3. **None, One** or **Multi** Objective Optimisation.
4. **Convex** Vs **Non-Convex** Optimisation.

5. **Non-linear** Vs **Linear** Optimisation.

6. **Deterministic Optimisation** Vs **Optimisation Under Uncertainly**.

An optimisation problem may belong to a combination of any of these categories, which essentially define the computational complexity of the examined problem. As mentioned earlier classifying an optimisation problem is important because it provides the necessary information for selecting an effective method for solving it.

1. **Continuous** Vs **Discrete** Vs **Mixed Integer** Optimisation—this classification is based on the domain of the input variables.

   Continuous Optimisation: the input variables can take any value from a bounded or unbounded set of numbers in $\mathbb{R}$. Continuous problems are found in economics, control theory, robot design and more. Examples of such problems are optimising the link lengths (continuous variable) of manipulators for reducing contact forces.

   Discrete or Combinatorial Optimisation—the input variables can take specific values, usually with a fixed separation distance, from a bounded or unbounded set of numbers. Special cases of discrete optimisation problems can be solved in polynomial time, but problems exist that belong to the NP-hard class. One of the most famous combinatorial problems is the *Travelling Salesman Problem* [77], and its decision version is an NP-complete problem. Given a list of cities and the distance between each pair of them, the salesman must find what is the optimal route to visit all of them once, and return to the city he started. Applications of such problems include: scheduling problems, routing problems (e.g., in the distribution of goods for determining the optimal route with the minimal distance that a delivery person should take), assignment problems (e.g., assigning resources to individuals) and more (see Doringo et al. [23] and Papadimitriou and Steiglitz [77]).

   Mixed Integer Optimisation—a subset of the input variables are restricted to be discrete values (e.g., integers) and the remaining variables continuous. These problems naturally arise in engineering due to discrete components. For example, to maximise the performance of a robot: (a) we can change the lengths of its links (continuous variable), and (b) select a DC electric motor out of a fixed set of off-the-self parts (discrete variable).

   Because this problem type has discrete decision variables, it is also a combinatorial type, and hence an NP-hard problem. This means that it cannot be solved by any known polynomial-time method. These problems are generally very tough to solve (Papadimitriou and Steiglitz [77]), and have recently received a lot of attention due to the many applications they find in engineering [93].

2. **Constrained** versus **Unconstrained** Optimisation.

   Constrained optimisation—is the problem of finding the set of all feasible solutions. As shown in Equation 2.1 there are two types of constraints:

(a) binding or equality constraints, which must be met exactly, and

(b) non-binding or inequality constraints, which are allowed slack.

Constrained optimisation problems commonly occur in many science disciplines and engineering. An example is a manipulator trying to grab an object. The manipulator can move its joints in a given range which are the joint limits (inequality constraints) and has to apply a specific amount of force (equality constraint) to grab the object without letting it slip or damaging it.

Unconstrained Optimisation—in these type of problems the goal is to find optimum values in all of the domain of the objective function f. In this case $F = \Omega$.

Except very simple problems most real-life problems are subject to constraints. A commonly used method for solving constrained problems is to formulate them as unconstrained problems. This happens by adding a penalty value to the objective function, which is equivalent to the amount of a constraint violation. More details on this are presented in Part II, where an explanation of a specific algorithm implementation is presented.

3. **None, One or Many Objectives**—optimisation problems can be classified also according to the number of objectives that they have.

   No Objectives—problems where there is no particular objective. An example of such cases are Feasibility Problems where solutions that are only required to meet the constraints are sought.

   One Objective—problems with only one objective, such as minimising the energy consumption of a robot.

   Many Objectives—also called multi-objective problems where several objectives exist, such as minimising the energy consumption of a robot while increasing its speed. For these type of problems the concept of Pareto front is used to determine the feasible solutions with the optimal trade-off. Naturally, the complexity of the problem increases with the number of objectives.

4. **Convex** Vs **Non-Convex** Optimisation, or **Quasiconvex** and **Non-Quasiconvex** optimisation for objective functions with multiple arguments.

   Convex Optimisation: is a sub-field of mathematical optimisation, where the problem has a convex objective function and its feasible set is a *convex set* (Boyd and Vandenberghe [12]). Specifically a function f is called convex when for each vectors x_1 and $x_2 \in \mathbb{R}^n$:

$$f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + f(x_2)(1-\theta),\ 0 \leq \theta \leq 1, \theta \in \mathbb{R}. \tag{2.6}$$

   In a convex problem there can be: (a) no feasible solution (unfeasible), (b) one optimal value. Convex functions are a special case of Quasiconvex functions. Most convex problems are tractable problems and have P complexity. Due to their simplicity and tractability they find many applications in several fields such as robot design, optimal control, function fitting and interpolation, and statistical estimation (Boyd and Vandenberghe [12]).

Non-Convex Optimisation—in this type of problem either the objective function or the constraints are non-convex. In non-convex problems there is either (a) no feasible solution (unfeasible), (b) only one optimal solution (that is also a global optimum), or (c) multiple local optima.

General non-convex problems might have several optima, making them harder to solve because in order to verify a global optimum all the local optima must be found and be compared. Convex problems have very convenient properties and are well understood making them easier to solve than non-convex problems ,and much more efficient in terms of time (Papadimitriou and Steiglitz [77]).

5. **Non-linear** Vs **Linear** Optimisation.

   Linear Optimisation—this classification is based on the mathematical model of the system. If both the objective function and the constraints are linear then the problem is also linear. These problems are a special case of convex optimisation problems and many of them, but not all, can be solved in polynomial time. Unless the objectives and constraints are linear, it is difficult to know if the problem is convex.

   Many problems in mathematics, economics, engineering, operations research and more can be expressed as linear problems by taking simplifying assumptions. For example, under the assumption that distances between destinations are straight lines (e.g., ignoring factors such as traffic lights, or turns etc.) 'the routing problem' (mentioned earlier in this list in the Combinatorial Optimisation category) of the delivery person becomes linear and can be efficiently solved. This assumption might not be realistic but can provide us with a solution that can work in most cases, and hence has a better practical use.

   The field that deals with linear optimisation problems is called *Linear Programming* and in general such problems are solvable in polynomial time. Because linear problems have one optimal value, they are faster and easier to solve than non-linear problems that may have multiple optima (Papadimitriou and Steiglitz [77]).

   Non-linear Optimisation—a problem is non-linear when the objective function or at least one of the constraints are non-linear. Non-linear problems can be non-convex meaning that they may have multiple local optima. Currently there are no efficient methods for solving the general non-linear problem (Boyd and Vandenberghe [12]), which in most cases is NP-hard.

   In nature there are many relationships that cannot be described by linear models. Examples of non-linear problems are: optimisation of non-linear chemical reactions or profiles of non-linear springs and dampers which are widely used for storing or dissipating energy in trains, cars, robots and more.

6. **Deterministic Optimisation Vs Optimisation Under Uncertainly**—this classification is based on the existence or absence of uncertainty in the problem.

Deterministic Optimisation—the output of the model is fully determined by the input. In general, it is a standard practice to simplify problems and their computational complexity by assuming that there is complete information on their model and inputs. Even though the assumption of determinism results in approximate solutions, in many cases it can provide useful results. This basically creates a trade-off between (1) accuracy when creating a sophisticated model, and (2) simplicity (deterministic models are inherently simpler).

Optimisation Under Uncertainly

Stochastic Optimisation deals with problems under *uncertainty*. In most cases however, complete information is difficult to obtain, and creating models that are very accurate is very time consuming and sometimes even impossible (see Ben et al. [7]). Noisy measurements, unmodeled factors that we do not take into account or even predictions about the future which we cannot be certain of are only a few examples of uncertainty. To tackle this issue random variables are used with specified probability distributions to characterise uncertainty, and the expected value of the objective function is optimised.

Robust Optimisation—the purpose of this field is to find solutions where changes in a small vicinity around them do not significantly change the output of the objective function [7].

One key difference between stochastic and robust optimisation is than in the former we assume that we know the probability distribution of the random parameters, while on the later we do not. Because we do not know the probabilities, in robust optimisation we optimise other measures, such as the *worst-case scenario*.

Robust optimisation is widely applied in manufacturing. If the objective function of a design is very steep at a given optimal point, then that point is not considered robust. The immediate consequence of this is that small manufacturing or modelling errors can result in significant degradation of quality of performance. A robustness study ensures the quality of a product does not degrade even in the presence of errors.

In some applications robustness is more important than achieving the physically maximum performance. In such cases the safest option is the best option. It is preferable to have a robust design that can perform consistently despite the uncertainties, rather than aim for the best outcome with a high chance of failure.

2.2 Optimisation Methods

In this section an overview of the different kinds of methods for solving optimisation problems is presented, together with a set of measures for making comparisons between them. The aim is to discuss how to select appropriate methods, and their criteria for their selection for solving problems in robot design.

2.2.1 Performance Measurements

These concepts help to understand the strengths and weaknesses of each optimisation method so that the most efficient algorithms for solving a problem can be selected.

1. **Computational Complexity of Algorithms**—defines how efficient the algorithm is in terms of resources. Important indicators are time and used memory for solving a problem.

   Time Complexity—execution time is commonly used as a measure of complexity. It is expressed as a mathematical formula that gives an approximation of the upper bound or *worst-case* scenario of execution time.

   The execution time of algorithms may vary for different problem instances of the same size so the worst-case scenario is used, which is the maximum time that an algorithm may require to produce a solution to a given problem. This can be expressed as a function of input to output; however, because this function is difficult to find, and due to the fact that the behaviour is mostly influenced by the factor with the highest order of magnitude the *asymptotic behaviour* of the complexity is used.

   Big Oh maps how execution time grows with the number of inputs. With this measure algorithms can be classified based on their worst-time scenario for solving a given type of problem. More formally: if f, g are two functions from $\mathbb{N}$ to $\mathbb{N}$, then $f = \mathcal{O}(g)$ if there exists a constant c s.t. $f(n) \leq c\,g(n)$, for a sufficiently large n.

   The common forms of $\mathcal{O}$ are the following and are presented starting from the easiest to the most difficult case.

   - $\mathcal{O}(1)$—trivial solution, has a fixed upper bound in time, regardless of the input size.
   - $\mathcal{O}(log\,n)$—non trivial but easily solvable in short time even for large number of inputs.
   - $\mathcal{O}(n)$: linear relationship, which is still easy and fast to solve.
   - $\mathcal{O}(n^2)$—quadratic relationship, more expensive but still manageable.
   - $\mathcal{O}(n^k)$—polynomial relationship, is expensive but sufficient knowledge exists for solving such problems. Problems that can be solved by a polynomial time algorithm are called *polynomially solvable*.
   - $\mathcal{O}(k^n)$—exponential increase of required steps with number of inputs. Very expensive to solve. Algorithms with this complexity are applied to difficult problems where small knowledge exists for solving them. Intractable for large input sizes.
   - $\mathcal{O}(n!)$—commonly known as the *brute force approach*. It is intractable for problems with many inputs. Algorithms with such complexity are applied to problems where the only known way to solve them exactly is trying all the possible combinations of solutions.

   Other Measures of Complexity—worst case analysis however is not the only factor that must be taken into account when comparing algorithms. One of the most famous cases to prove the

aforementioned statement is the *Simplex* versus the *Ellipsoid method*. Both methods can be used to solve linear problems and the latter is polynomial time, but the former is not. However, in practise the expected running time of some variants of the Simplex method is polynomial and on average better than the Ellipsoid method (Papadimitriou and Steiglitz [77]). For this reason other measures such as the *average-case* complexity must also be examined to have a more complete comparison.

In practice CPU-time is also used for evaluating a performance. This measure is hardware dependent and can be improved by: (1) buying more powerful computers, (2) using computer clusters and performing parallel computations, or (3) optimising the software (yes, optimise the software that optimises!) by using more efficient programming languages (such as C) or by minimising software communication overheads between platforms.

2. **Precision** describes how close a solution returned by an optimisation algorithm is to an optimal solution. Some algorithms may take a large amount of time or may never find an exact solution.

3. **Convergence** refers to how fast the algorithm can find optimal solutions. Convergence can be expressed in number of iterations or steps. Algorithms that cannot guarantee convergence may never terminate.

4. **Completeness** is the ability of a method to discover all the optimal solutions including the global optimum.

5. **Optimality** describes if a method can guarantee to find the single best optimal solution (global optmimum), out of the many optimal solutions (local optima). *Global convergence* is a combination of the convergence rate and optimality criterion which describes if an algorithm can guarantee to find the global optimum in some number of steps.

6. **Robustness** is the property of an optimisation algorithm to find the global optimum even when starting far away from it, without converging prematurely to a local optimum.

2.2.2 Classification

Based on the problem type, complexity, the input size and what we wish to achieve, an appropriate method and its implementation can be selected. Optimisation methods can be either:

1. Enumeration Algorithms,

2. Iterative Methods or

3. Heuristics.

Enumeration Algorithms are direct (or exact) methods with a finite sequence of instructions. This means that an algorithm requires a finite amount of resources (time and memory), is unambiguous and

terminates in finite number of well-defined steps. The algorithm guarantees completeness, optimality, and precision of the returned solutions (which are exact in the absence of rounding errors). As the name implies, enumeration algorithms discover all the optimal solutions, compare them and return the global optimum.

These properties make them easier and faster to apply; however, the most important problems have no known algorithms for solving them, or existing methods become intractable for large input sizes (e.g., are exponential time).

Iterative Methods are mathematical processes that start with an initial value, and produce a sequence of improving approximate values. These methods cannot guarantee convergence. However, if an iterative method has a termination condition it is considered an algorithm and can guarantee convergence at the cost of precision. Iterative methods are used to solve problems with many variables where direct methods are very expensive or intractable. Even with a termination condition, not all iterative methods can guarantee converge to an optimal solution, but provide a guarantee on the quality of the approximation (see Nocedal and Wright [75]). All iterative methods discussed in this thesis have a termination condition.

Heuristics (from the Greek εὑρίσκω which means I find or discover) are techniques to generate fast approximate solutions to a problem (not necessarily optimal), by guessing based on available information. Heuristics are also algorithms; however, they cannot guarantee convergence, completeness nor optimality. These methods are used when there is (1) no known efficient algorithm for solving a problem, and (2) the brute-force approach is intractable.

Heuristics sacrifice optimality, precision, and completeness for higher speed and are often used to solve NP-hard optimisation problems. The vast majority of Artificial Intelligence problems are solved via heuristics (see Russell [91]). Examples of heuristics include the widely used neural networks, which find applications in pattern recognition (e.g., voice or image) or agent behaviour (such as in video games).

Deterministic Vs Non-Deterministic Methods—the previously mentioned methods can be also classified based on the existence or absence of randomness in the sequence of steps the method follows.

- **Deterministic methods** provide the same output given the same input, computed from the same sequence of steps. These methods can guarantee optimality under certain conditions.
- **Non-Deterministic algorithms** are methods where given the input we cannot *determine* the sequence of steps nor the output. This means that different outputs can be produced for the same input.

  In algorithms the source of non-determinism could be due to the following.

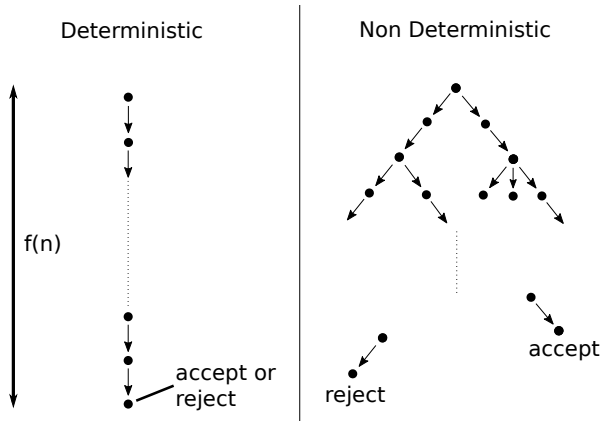

Figure 2.4: Deterministic (left) and non-deterministic decision algorithm. A deterministic algorithm will always follow the same steps and terminate in finite time. A non-deterministic algorithm does not necessarily follow the same steps nor produce the same output, and might never terminate.

- A random external signal other than the input, that follows an unknown probability distribution.
- Time-sensitive operations, for example in multi-threading machines that choose the next step to execute depending on which thread will finish first. Such concurrent processes are dependent on *race* conditions, which in general cannot be predicted.

Stochastic Methods are methods that use random variables with a known probability distribution to determine each step of the algorithm. By using probabilities a faster exploration of the search-space is achieved. These methods can handle practical and large scale problems. Stochastic and non-deterministic methods do not mathematically guarantee optimality, but trade this disadvantage with speed, and are able to provide reasonable solutions.

Metaheuristic is an algorithmic procedure that generates heuristics for solving problems. They are general-purpose algorithms that can be applied to a variety of different optimisation problems without the need of large modifications. Many metaheuristics use stochastic optimisation techniques. Metaheuristics are used when there is incomplete information of the examined problem. An example of algorithm that is using a metaheuristic is the family of Evolutionary Algorithms (EA) [18], which will be extensively discussed in the next part of this thesis. Evolutionary algorithms are specifically interesting because they are based on the real phenomenon of evolution.

2.2.3 Techniques

Many optimisation methods are tailored for solving specific problems. In the recent years though, a lot of work has been made towards developing all-purpose methods that can be applied to a variety of problems. Because there are many different techniques for solving optimisation problems, only the most popular and relevant to robot design are shortly presented and described.

Exploration Vs Exploitation

Before presenting the various techniques two basic concepts must be presented first. The concepts of *exploration* and *exploitation*.

Exploitation is a low-risk strategy that seeks to maximise the use of available information. Strategies that seek to purely exploit information are called *greedy strategies*.
Exploration is an approach where new information is sought at the cost of not fully exploiting the current information. These strategies seek to discover new information even when moving in a direction where the objective values is decreasing.

Obviously these two terms are conflicting. This creates a trade-off that global algorithms must optimise. If a new solution has the highest overall value then the algorithm must search the area around that solution to find the locally best one (exploitation). However, if that solution is not close to the global optimum, the algorithm might waste all of its resources to find a sub-optimal solution. For that reason, new solutions must be constantly explored (exploration), because as mentioned earlier the algorithm can never know for certain (unless it explores the entire search-space) if a given point is a global optimum or not.

Exploration is mostly governed by the philosophy of taking blind or random steps in hope of finding a better solution. The trade-off between exploration and exploitation in optimisation algorithms can be regulated via fine-tuning of their hyper-parameters or by selecting different implementations of a method. These parameters and implementations may vary between methods, and in many cases there are specialised implementations that are specifically tailored for promoting *diversity* between examined points. Furthermore, the algorithm's hyper-parameters can tilt the scale towards exploration or exploitation. For example, in the case of Genetic Algorithms (Section 2.2.3), a high probability of genetic mutation leads to stronger exploration.

Following the previous explanations, optimisation algorithms can be classified into two general categories:

1. global optimisation algorithms, and
2. local optimisation algorithms.

Global Optimisation

Global optimisation algorithms are used for solving problems that are expected or are proven to have multiple optima. Because such problems are usually intractable, heuristics are used for finding fast solutions. To increase the chances of finding the best solution, global optimisation techniques evaluate multiple points that are scattered over the search-space. To generate a population of initial solutions, methods that are referred to as *Design of Experiments* (DOE) are used, in an attempt to promote exploration. These techniques will be described extensively in Part II. Because these algorithms can

escape from local minima, they can be applied in most problems. Examples of global optimisation algorithms are the following.

- **Evolutionary Algorithms** (EA) are a family of metaheuristic global optimisation algorithms that are inspired by evolution and the process of *natural selection*. Starting with an initial design population, the *fitness* of each design is evaluated based on a performance objective criterion, and the best designs are selected to *reproduce* and produce new generations. Each design is encoded as a *vector of values*, and breeding happens via the basic genetic operators, which are the following.

  - *Selection*—the fittest designs are selected and propagate unaltered to the next generation.
  - *Mutation*—based on a probability distribution some values on a design's vector of values are randomly changed.
  - *Crossover*—two of the fittest designs exchange genetic information, which in practice means a new design is generated by combining the genes of the parents.

  Based on the aforementioned procedure, several variations of EA were developed for solving generalised optimisation problems. Some of the most popular techniques are *Genetic Algorithms* and *Evolution Strategy*. Due to the fact that these algorithms use metaheuristics they cannot guarantee convergence, completeness nor optimality; however, because they can produce fast solutions and can be easily applied, they have been widely used in robot optimisation. An overview of evolutionary-aided robot design can be found in Prabhu et al. [82] and more details about the theory behind EA can be found in Coello [18].

- **Particle Swarm Optimisation** (PSO) is a metaheuristic technique based on animal flocking behaviour. Starting with an initial population of designs scattered in the search-space, particles (the designs) start moving towards a locally best known design. By doing this, it is expected that the particles converge to the best values. This technique is similar to EA, in the sense that it uses the concepts of population and fitness measurement [18]. Just like EA, PSO takes few or no assumptions about the examined problem, making it easy to apply; however, just like any other metaheuristic it does not provide any guarantees.

- **Simulated Annealing** is a stochastic metaheuristic optimisation technique that mimics the process of *annealing* in metallurgy [18]. Based on a temperature value, the algorithm seeks new values, even if they are worse than the current best solution, while the temperature gradually falls. As the temperature lowers, the probability of accepting worst solutions also drops. This allows the algorithm to jump from local minima to new areas in search of a global minimum.

- **Branch and Bound** (BB) method is a divide and conquer method for solving discrete optimisation problems. It splits the search space into smaller spaces (branch) and *prunes* tree branches based on calculated bounds on the objective function value (Papadimitriou and Steiglitz [77]).

Without the bound part this method would be an exhaustive search. This method does not guarantee optimality nor convergence and has an exponential worst case complexity, even though it can produce acceptable results in practice.

- **Artificial Intelligence** uses statistical models for approximating solutions of general problems. The basic concept of AI is to *train* statistical models to optimise a cost function. This means that most AI methods are stochastic. Examples of AI optimisation methods are *Bayesian Optimisation* (BO), variants of *Artificial Deep Neural Networks* (NN) and Reinforcement Learning (RL). BO uses a distribution (prior) to model the objective function and the distribution is updated with each new sample (posterior). NN and RL are commonly used to model control strategies by making control decisions to optimise an objective (e.g., avoid collisions or generate a walking pattern). More details on AI and its applications can be found in Russel and Norvig [91].

Other noteworthy global optimisation algorithms are the *Tabu Search* [18] and the *Ant Colony* (Doringo et al. [23]).

Local Optimisation

Generally referred to as *gradient-based* or *classical* approaches, local optimisation techniques search for the best possible value close to a starting point, meaning that they are fitting methods for exploitation. These methods use gradient information and usually come with guarantees about optimality and fast convergence rates, and provide solutions with high precision. However, they do not provide completeness (because they return only one solution) and can easily get trapped in local optima. This happens because the result is dependent on the initial value. These methods are most commonly used for linear or non-linear convex problems, but can also be applied to *refine* approximate solutions returned by global optimisation techniques.

Local methods depend on derivative values for calculating an iterative process of steps. If derivative information is not available, it can be approximated using *Forward* or *Central Differences*, or any other methods for approximating derivatives (see Nocedal and Wright [75]). These methods have multiple variations so the most general cases are only briefly described.

- **Gradient descent** is an iterative method for non-linear and differentiable problems. The intuition is simple, move in the direction where the objective decreases with the fastest rate until a critical point is reached, that is *hopefully* a local minimum. The update rule is given by Equation 2.7, and it is commonly used with a step value $\gamma \in [0,\ 1]$.

$$x_{i+1} = x_i - \gamma \nabla f. \tag{2.7}$$

This method is often used in the training process of Artificial Neural Networks (Russell [91]).

- **Newton's method** is an iterative method for unconstrained non-linear and twice continuously differentiable problems. It requires information of the first and second derivatives of the objective function (Jacobian and Hessian for multivariate functions), and every new step is calculated by Equation 2.8, which is derived from the Taylor series expansion. For this reason the second derivative needs to iteratively be calculated (and its inverse) to iteratively solve a linear system of equations. The difference with gradient descent methods is that Newton's methods use curvature information to reach at the optimum faster, but require the second derivative to exist and be continuous. For a scalar value x, the step can be calculated as follows:

$$x_{i+1} = x_i - \gamma \frac{\nabla f}{\nabla^2 f}. \tag{2.8}$$

  If information of the derivative is not known, or difficult to estimate, *Quasi-Newton* methods can be used instead. Their advantage over *exact Newton's method* is that they do not need to calculate the inverse of the Hessian, which is very expensive, but they approximate it. Quasi-newton methods are computationally cheaper and faster than newton's methods, but they have a slower convergence rate and are less precise (see Nocedal and Wright [75]) because they do not have complete information of the second derivatives. Newton's method is rarely used standalone since most problems of interest are constrained. One of the most popular algorithms that use the quasi-newton method is the Broyden–Fletcher–Goldfarb–Shanno (BFGS).

- **Simplex algorithm** is a method for linear optimisation problems. A system of linear inequalities creates a polytope, and the algorithm starts from a vertex and moves to a next vertex through the edges until the optimal solution is found. Simplex has an exponential worst-case complexity; however, in practise it has a polynomial-time average complexity making it a powerful tool for linear optimisation problems (Papadimitriou and Steiglitz [77]). Simplex algorithm has many different variations.

- **Sequential Quadratic Programming** (SQP) are iterative methods for constrained and twice continuously differentiable problems. It is suitable for problems with significant non linearities. SQP methods turn large non-linear problems into smaller and easier quadratic problems, and use newton's method to iteratively solve the *Lagrangian function*. For this reason it can be very efficient in large scale problems (Boyd and Vandenberghe [12]). SQP methods find uses in both robot design and optimal control problems. Because of their efficiency they can be used as parametric controllers for real-time non-linear control problems.

- **Interior Point** (IP) methods are iterative methods for solving linear and non-linear convex and constrained optimisation problems by solving a sequence of approximate optimisation problems. The approximate problems are equality problems and are easier to solve than the original problem [12]. As the value of the approximate problems goes to the minimum the value of the objective function *should* go to its minimum. In contrast to Simplex, instead of

traversing edges IP algorithms traverse the interior of the feasible region until a solution is found. They are a popular derivative of the newton's method and they iteratively approach the optimal solution from inside of the feasible set. The most common methods are the *Barrier* and the *Primal Dual* methods. They have been proven to be polynomial-time solvable for linear problems. IP methods are very popular and *Matlab's Optimisation Toolbox* [67] contains an implementation.

Directly obtaining derivative information is quite difficult for most problems, and hence derivative approximations via Finite Differences are commonly used. The aforementioned methods should be applied when the examined problem meets regularity conditions (e.g., objective function is differentiable) otherwise convergence is not guaranteed.

2.3 Conclusion and Summary

In this chapter an overview of mathematical optimisation and its basic concepts was presented. In the first section a formal definition of a general optimisation problem and its most important concepts are explained. Thereupon, a short introduction to computational complexity theory is presented, which is essential for understanding why some problem types are more difficult to solve than others. Furthermore, a classification on the various types of optimisation problems is proposed together with some important examples of each category's problems.

Next, a set of performance criteria and the different types of methods for solving optimisation problems are presented. Finally, the chapter ends by briefly describing some of the most commonly used optimisation methods for robot design and a short discussion on when they can be used.

The purpose of this chapter is to lay out the foundation for evaluating the pitfalls of robot design optimisation, as well as to present the criteria for selecting appropriate methods. In the next chapter, applications of numerical optimisation for the design of robots is presented together with a critical assessment of the field.

Chapter 3

Related Works: Robot Design Optimisation

Robot design optimisation is the field that deals with finding the best design to perform a set of tasks, based on a set of requirements. The objective of this chapter is to situate the presented research in the literature. More specifically, to present what has been written on the topic of robot design optimisation, the main approaches, and the gaps and weaknesses that this study helps to fill.

The process of designing a robot requires experience and an understanding of the non-intuitive relationships between the desired behaviours and the mechanical parameters of the robot. To facilitate and understand this process, researchers have turned to computational optimisation. Robot design optimisation is a relatively young field. Some of the first notable results in the field were presented in the work of Sims [101] in 1994 where an Evolutionary Algorithm was used for co-optimisation of motion and physical structure to create virtual creatures that could jump, walk or swim.

Robot design optimisation deals with problems that exist in the real world, so optimisation studies may involve a large number of parameters and constraints, as well as multiple objectives. Furthermore, many components of a robot are non-linear, and independent variables can be continuous or discrete. As a result, the majority of such problems are computationally intractable (NP-hard), which means that we cannot expect to solve them efficiently or even to find exact solutions with optimality guarantees. For this reason scientists have devised different ways for designing robots, including trial and error, with optimisation tools or even by mimicking the designs and behaviours of animals.

This chapter starts by presenting the bio-inspired design approach, where concepts from nature are applied to the design of robots. Next, studies that optimise robot designs for achieving a set of predetermined motions or behaviours are presented. Following this, studies where the design is predetermined and an optimal motion or controller is sought are presented. Finally co-optimisation studies of design and the way to achieve the desired motions are discussed. For completeness, design studies of various types of robots are presented with a main focus on legged robots, which is the case study of this thesis.

Figure 3.1: From left to right: (1) Centauro robot, (2) MIT's mini-cheetah, (3) Hyq2Max.

3.1 Bio-Inspired Robot Designs

Before presenting numerical optimisation approaches, it is worth mentioning the category of *nature-inspired* or *biologically-inspired* robots. Instead of relying on computational models or pure intuition, scientist attempt to copy the physical characteristics of animals such as their skeletal or muscle mechanisms. Compared to the few years of human advancement in technology (air planes were invented only a century ago), nature has spent billions of years of evolution to eliminate weaknesses and produce stronger species. The amazing abilities that animals display are the result of nature's own optimisation via the process of evolution.

Many animals have very unique designs and have developed very efficient strategies for moving based on their inherent evolutionary traits. An interesting study on how animals move is presented in the work of Dickinson et al. [21]. Scientists have based robot designs on a variety of animals in an attempt to reproduce some of the amazing feats that these animals can achieve. Scarfogliero et al. [95] designed a miniature robot inspired from frogs, that was capable of achieving travelling hops of 1.5 m/s. Murphy et al. [73] designed a quadruped robot inspired by dogs, with the aim to explore legged-locomotion in different terrains. The robot was designed for learning and experimentation, and performance requirements were not clearly stated in their work.

Quadruped robots are popular due to the stability provided by their four legs, which can help them balance without effort (Raibert [83]). Similarly, Semini et al. [98] designed a quadruped robot named *HyQ2Max* (see Figure 3.1) based on bio-mechanical studies on humans and animals. Even though the authors perform an optimisation study based on torque requirements for minimising actuator sizes, and four-bar linkage parameters, they state that the design does not have a specific application. Resembling also a dog, Hutter et al. [52] designed a general purpose robot named *Anymal* with a series elastic actuator inspired from biology, which was able to demonstrate a variety of behaviours such as trotting, walking and stair climbing.

Cheetahs are also a source of inspiration. These quadruped animals are the fastest land animals on earth and can reach speeds of up to 130 km/h. Sprowitz et al. [105] designed a small quadruped robot called *Cheetah-cub* for demonstrating trotting gaits; however, no specific performance requirements were defined in their work. Inspired also by cheetahs, a series of robots were build by the Biomimetic Robotics Lab of MIT. Bledt et al. [9] presented the *MIT Cheetah 3* intended for general use, demonstrated by its ability to blindly climb stairs. Following the larger prototype of Bledt et al. [9], Katz et al. [56] built *mini-cheetah*, a smaller and lighter version of the Cheetah 3 capable of performing a back-flip.

Inspired from the centaur creature from the Greek mythology Kashiri et al. [55] designed the *Centauro* robot (see Figure 3.1), that has four legs and the upper body of a humanoid, in an attempt to utilise the stability provided by four legs and the manipulation capabilities of humanoids for disaster applications. Graihen et al. [40] designed a kangaroo with the aim to achieve stable hopping cycles; Chablat, Venkateswaran and Boyer [14] created a caterpillar-based piping inspection robot to efficiently navigate inside pipes; and Kuehn et al. [59] designed a chimpanzee-inspired robot. Finally, Li et al. [62] and Yim et al. [111] demonstrate hopping behaviours in insect-like robots.

Commercial legged robots outside of industrial environments are already starting to appear by the time this thesis is written. Even though their design details are not revealed, they are worth mentioning as they are an indication that robotics has reached a level of maturity where reliable products for general-purpose use are already produced and sold. These robots provide an insight into the future where robots will be a part of our everyday lives, and they already find applications in military, exploration, disaster scenarios and even advertisements.

Introduced in 1999, the *AIBO* robot [103] is a dog-inspired robot created by SONY for entertainment purposes, which was also used for educational and academic purposes. One of the most prominent companies in robot manufacturing is *Boston Dynamics* with the *Spot Robot* [25], a commercially available quadruped. *Unitree Robotics* [86] and *Ghostrobotics* [85] are also ready to enter the market with a series of agile quadruped robots. *Agility Robotics* [84] already produces and sells bipedal robots intended for in-home use.

3.2 Design-only Optimisation

Design-only optimisation is concerned with studies where only the design of a robot is optimised to achieve a set of tasks in a predetermined way. In other words, the optimal design for achieving a specific motion or behaviour is sough via a numerical optimisation experiment. This type of optimisation is usually performed when the goal is to design general-purpose robots or when behaviours are difficult to model and simulate. The following categories can be defined for design optimisation.

1. **Topology optimisation**—are techniques that are not bounded by predetermined structures. Can be used for (1) maximising the rigidity or reducing the mass of a structure, or (2) for

applications where different shapes can produce substantially better results (e.g., for the design of soft robots such as Hiller et al. [50] and Cheney et al. [17]). The former case usually requires stress-strain analysis to strengthen areas of high strain and shave off mass from areas of low strain, which is usually performed with Finite Element Analysis. A drawback of this analysis is that it is very computationally demanding, because it applies mesh generation techniques to create finite elements for solving a set of partial differential equations.

2. **Shape or morphology optimisation**—the shape has a predetermined structure to be optimised and is described by a parametric function. In addition to the properties of the shape (e.g., length, curvature etc. [92, 46]) the behaviour of a component or the entire robot can also be optimised by varying parameters that represent selected properties (e.g., stiffness of a spring).

3. **Mixed-integer optimisation**—in this case specific components can be replaced. For example, in the case where the performance of different actuator systems needs to be tested. Various actuators have different transfer functions, and hence changing this component changes the behaviour of the entire system.

In the work of Ha et. al. [44], an optimisation approach for generating robot designs for tracking a set of input trajectories is presented. The approach uses discrete components such as actuators and mounting brackets to create optimal robots for tracking reference trajectories on manipulators and legged robots. The work is validated by demonstrating its effectiveness in fabricated robots.

Chablat, Venkateswaran and Boyer [14] minimised the size and maximised the contact forces for a piping inspection robot to efficiently navigate inside pipes by using a Genetic Algorithms. The robot is optimised to mimic the motion of a caterpillar. Ceccarelli and Lanni [13] optimised the geometrical parameters of a simulated 3R manipulator with multiple objective functions to increase its workspace capabilities using a SQP optimisation method, which is a local optimiser.

Semini et al. [98] optimised hydraulic cylinder sizes and the kinematic parameters of a 4-bar linkage of *HyQ2Max* to fit a desired output torque profile. The trajectories come from a set of simulated motions on a previous version of their robot. Even though this approach could provide sufficient information for creating a robust new design, there is no evidence or study showing that these motions are optimal for different designs; as it is shown in Section 6.6.7, designs with small variation in their parameters may present a significantly different performance. Roozing, Zeyu and Tsagarakis [87] optimise spring stiffness and positions of a monopedal robot for standing on a flat terrain.

In the work of Yim et al. [111] high-performance hopping and balancing ability is demonstrated with a miniature hopping robot named *Salto-1P*. The performance of their miniature hopping robot is measured via experimentation in a series of tests; however, this method cannot always guarantee that the desired performance would be reached. Plecnik et al. [79] optimised the parameters of the Salto's 8-bar linkage for producing jumps with almost zero angular momentum. Even though a set of constraints and optimisation objectives were defined to produce low angular momentum hopping

behaviours, these behaviours were not simulated. The optimisation was performed using the Interior Point Algorithm, a local optimiser.

All of the aforementioned robots demonstrate impressive skills, and were built with a clear general sense of purpose such as to perform accurate hops or have high reachability in the case of the manipulator; however, their goal was not explicitly defined (e.g., define exactly the desired hopping height), and their physical capabilities were mostly determined experimentally without any proof that the final design is the best one for the desired objective. Different designs have different abilities, and behaviours must be adapted to properly exploit them. This issue is discussed in Section 6.6.

3.3 Motion or Behaviour Only Optimisation

In this section optimisation studies are presented where optimal behaviours of a given design are sought. Such studies may be either on real robots or for the purpose of producing realistic animations. This type of study usually precedes the design of the robot, and can be divided into three main categories.

1. **Motion or trajectory optimisation**—in this case a signal is modelled and is directly optimised. The advantage of this method is that it gives freedom to the optimiser (because it is not restrained by the capabilities of a predefined controller), and hence it has more chances of finding better solutions. The disadvantage is that once an optimal signal is discovered, a controller must be built based on it, and this might turn out to be a tedious task. This is a case of *open-loop* control, where the system does not use *feedback* to determine if the commanded signal has achieved the desired goal. The control system is being ignored and only the plant is being investigated.

2. **Behaviour optimisation**—this case includes the motion optimisation (e.g., optimises a given signal) as well as certain key variables (e.g., initial position of joint and velocities). In other words, both the initial conditions and the action that the system takes are optimised, which essentially forms a full behaviour. This category gives the highest degree of freedom to the optimiser, but results in more complicated problems due to the higher number of independent parameters. Naturally, this is also an open-loop control case.

3. **Controller optimisation**—is performed when the system already has a controller and the parameters or policy of the controller are optimised to achieve the best behaviour. The advantage of this approach is that less effort is required because no new controller needs to be designed; however, the system can be constrained by the nature of the controller. A controller can be parametric (e.g., PID controller), an optimisation algorithm (e.g., SQP controller) or a heuristic (typically Artificial Intelligence such as Deep-Neural Networks or Reinforcement Learning [91]). In this case *closed-loop* behaviours are investigated and the control system is being taken into consideration.

Given a specific design, various behaviours can be achieved based on its characteristics (e.g., it has legs, wings etc.), and its physical traits. Sometimes, these behaviours might include unfortunate scenarios such as loosing a limb. In an interesting study Bongard et al. [10] investigate behaviours where a quadruped robot adapts its locomotion strategy when amputated, by using optimisation techniques. Such behaviours can increase the robustness of robots and allow us to get a deeper understanding in the cognition of animals, so we can integrate it into machines.

Sprowitz et al. [105] explore the physical potential of the cheetah-cub for trotting gaits using Particle Swarm Optimisation (a stochastic global optimisation technique) for optimising control parameters. Bledt et al. [9] explore a variety of behaviours including stair climbing by optimising contact forces or robot states by assuming linear dynamics with Quadratic Programming (QP). This optimisation approach is dependent on initial values and is prone to getting trapped in local minima. On the same robot Nguyen et al. [74] explore jumping behaviours by optimising trajectories (joint states and torques) also with QP, and searching only for a feasible solution. Even though they demonstrate hopping behaviour on and off a 0.76 m table, they present only one way to achieve it, and there is no certificate of optimality.

Katz et al. [56] use an implementation of the Interior Point algorithm for generating a behaviour to achieve a back-flip of 0.3 m height on MIT's mini-cheetah. In their work, they treated the optimisation problem as a feasibility problem and searched for any possible way to achieve this feat. Searching only for feasible solutions means that the presented behaviours are potentially non optimal. On the same line, Graihen et al. [40] use Matlab's [68] non-linear optimisation solver for generating feed-forward stance phase trajectories of the joints for landing/lift-off in their kangaroo-like robot.

The computer graphics community has also demonstrated a plethora of studies in design optimisation. These studies are mostly restricted to behaviour studies such as in Fang and Pollard [28], where the authors optimise the behaviour of humanoid animations to perform a variety of complex dynamic motions such as back-flips. Macchieto et al. [66] optimise joint accelerations using QP for balancing humanoid and non-humanoid 3 D animations.

The main disadvantage of behaviour-only optimisation is that the physical capabilities of the robot or the simulation have been already defined and such studies can only determine the optimal behaviour of a specific design. A different design might display a higher performance for a given task, but it is impossible to determine that without examining how various designs perform on a given task. Furthermore, most of the previously mentioned studies were feasibility studies or were performed using local optimisers. As a result, only a single solution or strategy is found, and other local minima (and probably the global optimum too) might be ignored. Certainly physical demonstration in a real robot provides a proof of concept and is useful for future studies. However, to design robots that are reliable and perform to their maximum potential, studies should include global exploration which can discover multiple solutions and allow the design to co-evolve with its behaviours.

3.4 Design and Behaviour Co-optimisation

This thesis argues that explicitly specifying the performance objectives of a robot and co-optimising both the behaviour and the design can lead to more robust, energy efficient and capable robots. Understanding the purpose of the designed robot can provide additional information that can aid the design process and result in impressive behaviours. For example, even though the performance objectives were not explicitly stated (e.g., achieve a 1 m hop), mini-cheetah [56] and Salto-1P [111] were designed to be robust and display highly dynamic jumping behaviours, which they do. This indicates the importance of a more specific *design purpose*. Taking this a step further, one can explicitly define one or multiple objectives for the designed robot. In this way, the designer can investigate which are the limitations of his design, based on the simulations, and improve it.

A robot's performance can be limited by the potential of the available technology, and without exploring it in a systematic way we can not achieve what is truly possible. Robots should be designed to meet a clear set of performance objectives, and not be experimentally evaluated afterwards, which might lead to sub-optimal behaviours. This section presents and evaluates optimisation studies in which performance objectives are explicitly defined and are concurrently optimised with the designs.

Some of the first notable works in design and behaviour co-optimisation are presented in the work of Leger et al. [61] (1999). In this study, the author implemented a multi-objective platform for design and controller co-optimisation. The aim of this work was to introduce a generalised framework to automate the design process of various types of robots for defined task objectives. In another early work of Lipson and Pollack [64] (2000), robotic structures of various shapes are co-optimised together with their controller for the task of locomotion.

Gradient-based approaches have been widely applied in design optimisation due to their fast converging properties, but require carefully selected starting points and are constrained to a local exploration around them (they cannot escape local optima). Spielberg et al. [104] co-optimised motion and morphology of legged and flying robots for one task using a gradient-based approach. In this study the authors built and tested a real hexapod in which optimisation of link lengths was performed for the task of running.

Evolutionary algorithms have also been extensively applied for the problem of design and behaviour co-optimisation due to their global convergence properties. Surveys and reviews, where the key challenges on the field are presented and discussed can be found in the works of Winfield and Timmis [110], Bongard [11], Gupta and Singla [43] and Prabhu et al. [82].

3.4.1 Soft Robots

Design and behaviour co-optimisation has also been the topic of interest in soft robots. Soft robots are challenging to optimise due to their compliant bodies, which are difficult to model. In model-based approaches models are built and optimised in simulations; while model-free approaches use real robots for evaluation. The rapid evolution of additive manufacturing has facilitated the automation of the fabrication process of soft robots, which results in creation of soft robots faster and with minimal

human intervention. A review on model-free and model-based co-evolution of soft robots is presented in Howison et al. [51], together with a new emerging trend that combines the two approaches.

In the work of Vujovic et al. [108] a model-free approach for co-optimisation of soft robots for locomotion is presented. In this work a manipulator is used to automatically assemble and evaluate new robots with a webcam. Using the same philosophy Rosendo et al. [88] automate the fabrication process for co-optimisation of soft robots. Model-free approaches and validation of multiple design candidates in the real world is a very interesting approach; however, it is not easy to implement for larger robots with complicated mechatronics.

A study that focuses more on the theoretical insights behind EA can be found in [58], where a simulated soft robot's control and morphology are optimised for locomotion. In this study, soft robots are represented by voxels with a varying size. Another interesting study on soft robots is presented in Cheney et al. [16] where virtual soft robots driven by current are co-optimised for locomotion in flat ground using EA. In a recent study by Cheney et al. [17], a new approach in EA is presented and is applied to co-optimisation of topology and control of virtual creatures with the aim of giving a deeper insight in embodied cognition.

Baykal and Alterovitz [6] use the Adaptive Simulated Annealing algorithm (ASA), a global optimisation algorithm for optimising kinematics of a simulated cylindrical surgical robot to navigate in cluttered environments. Motion is generated by a sampling based algorithm named Rapidly-exploring Random Tree (RRT), to generate approximations of a design's reachability by randomly searching for collision-free paths.

3.4.2 Manipulators

Hazard, Pollard and Coros [48] co-optimise the kinematics and trajectories of manipulators for different manipulation tasks using the Broyden Fletcher Goldfarb Shanno algorithm. The feasibility of their approach has been proved on a 3 D-printed hand that grasps a sphere. Similarly, Chen, He and Ciocarlie [15] co-optimise the design and the control parameters of an under-actuated hand for grasping objects using Reinforcement Learning techniques. Their approach is validated with a real robot that successfully grasps different objects. Ha et al. [46] optimise link lengths and trajectories for moving and picking objects with manipulators, a work which is more extensively discussed in the following subsections.

3.4.3 Other types of Robots

Mechanism and behaviour co-optimisation has also been applied in micro-robots. Liao et al. [63] use a variation of Bayesian Optimisation to optimise design and controller behaviour of micro-robots for locomotion. These robots are in the millimetre scale, and weight only a few grams. In the work of Luck et al. [65] the controller and morphology for traversing terrains with sand is co-optimised. Locomotion is inspired by sea-turtles and the approach is validated with experimental results. Corucci et al. [19] optimise the design and the feed-forward behaviour of an underwater robot, and Moore et

al. [71] optimise the controller and the design of an amphibious robot with EA, and validate their approach on a real prototype.

3.4.4 Legged Robots

The design of a humanoid robot together with walking trajectories and a controller's feedback gains are co-optimised in the work of Cruz et al. [20] with the use of genetic algorithms. The results are validated in a real humanoid robot. In the work of Zhao et. al. [112] robot assemblies are generated and their controllers are co-optimised for traversing various terrains. A heuristic algorithm is used to generate virtual multi-legged robot designs from a database containing discrete predefined links, joints and connectors, and Model Predictive Control is used for generating optimal trajectories. Their simulations include a maximum joint torque-limit, which corresponds to the limit of an actuator they already have; however, they do not use a complete model of an actuator, and hence its dynamics are ignored.

Pathak et al. [78] co-evolve control and morphology of virtual robots that consist of limbs and motors for standing up. The robots learn to self-assemble to form more complicated robots and optimisation is performed with Neural Networks. Geilinger et al. [37] co-optimise design and motion for legged robots that also have wheels on their feet. For optimisation they use Newton's method with a backtracking line search method (a local optimisation method), and they validate their results on real robots. Ha et al. [45] co-optimised the design and motion of a virtual single-legged hopping robot to reach a 1 metre vertical hop. With the objective of maximising gait stability and speed, Nygaard et al. [76] co-optimised morphology and control in a real small quadruped.

In their work, Saar et al. [92] used a Bayesian optimisation algorithm for co-optimisation of morphology and controller. Their study was applied to a real hopping robot for optimising running speed and a few design parameters including curvature of the foot. Schaff et al. [96] used a combination of a Gaussian-Mixture model (a Bayesian optimisation technique [91]) and deep reinforcement learning for co-optimising control and morphology. In their work, simulated legged robots, which included a hopper, were optimised with the aim to maximise their horizontal locomotion speed.

In a recent study by Ha et al. [46], the authors propose a generalised framework applicable to different kinds of legged robots and manipulators that receives as input a parametrized robot with a motion trajectory, and concurrently optimises design and motion. Their motive was to aid the design process by modelling the relationship between design and motion via sensitivity analysis. Sensitivity analysis is very important for understanding the examined system, and it is an integral part of the proposed design approach in this thesis.

The framework of Ha et al. [46] was applied for the optimisation of different robots and different tasks. The design parameters optimised are link lengths, position of the actuators and geometrical characteristics of four-bar linkages. In the motion trajectory part, the optimised parameters are actuator and contact forces as well as joint positions. In their work the motions are given, whereas in the work presented in this thesis the objectives of the behaviours are provided together with initial values,

allowing the optimiser to find the optimal way to achieve them. Finally, their results were applied on a real miniature quadruped robot for rolling and walking gaits.

3.5 Comparison With Other Works

This section focuses on the gaps in the works presented for the field of robot design optimisation, and how the presented research helps to fill them and add to the current knowledge.

Advantages of Design and Behaviour Co-optimisation—in the presented framework both the mechanism and the behaviour of the robot are co-optimised, which allows the design to adapt to new behaviours and vice versa.

Several studies optimise a behaviour given a specific design, such as in Bledt et al. [9] and Verstraten et al. [107]; however, this approach ignores the inextricable relationship between a design and its behaviours, which might potentially lead to sub-optimal results. In a co-optimisation study, the performance of a behaviour is not constrained to the design's maximum physical ability nor the design to a potentially sub-optimal behaviour.

Realism of Models—despite a few cases such as the work of Spielberg et al. [104] and Digumarti et al. [22] (which include realistic actuation constraints such as torque limits), most studies ignore important aspects of real robots which are taken into account in this thesis. Namely a few of them are: the dynamics of actuators and transmissions, physical limitations of the mechanism such as torque, current and speed limits, as well as realistic models of springs and their behaviours (such as hysteresis). As it is shown in the examined case study (see Section 6.6), all of these aspects play an essential role in determining a real robot's physical capabilities, and their effects and dependencies have not yet been properly explored.

Optimisation for Multiple Performance Objectives—most of the previously mentioned studies optimise for a small set or even only one desired behaviour. However, the framework presented in this thesis can be used for a large repertoire of performance objectives.

Ha et al. [45] co-optimise the design and motion of a monopod hopping robot for a one metre vertical hop; Nygaard et al. [76] co-optimise morphology and control in a real small quadruped. One behaviour is co-optimised in Digumarti et al. [22], and in Ha et al. [46] co-optimisation is performed for a variety of robots for one or two behaviours. As it is shown in the results of this thesis, achieving multiple different behaviours is not straight forward, due to the underlying trade-offs between them. For that reason, and in order to design robots that are useful and capable of multi-tasking, different performance objectives must be taken into consideration during the design process.

3.6 Conclusion and Summary

This chapter presented related works to this thesis topic, which is robot design and behaviour co-optimisation. The bibliography was divided into four parts, which are: (1) bio-inspired robot designs, in which robot designs are based on animals, (2) design-only optimisation, where robot designs are optimised for achieving specific tasks in a predetermined way, (3) behaviour-only optimisation of a given design where the maximum physical potential of a robot is sought, and (4) design and behaviour co-optimisation where the design and the behaviour of robots are concurrently optimised. Finally, a short description of the contributions of this thesis and comparisons with previous works was discussed.

In the next chapter the main contribution of this thesis is presented, which is a framework for design and behaviour co-optimisation of robots, as well as a proposed optimisation methodology for performing a design study.

Chapter 4

Optimisation Framework and Methodology

Abstract

This chapter introduces the main contribution of this thesis, which is an approach or philosophy for the design and behaviour co-optimisation of high-performance mobile robots. The approach consists of an optimisation framework and an optimisation methodology. The framework is a conceptual structure that serves as the canvas for the design study. The main intuition behind the framework is that a mechanism and its behaviours have an inextricable relationship, which is taken into account during the design stage, and together with a set of important concepts provides a foundation for the design optimisation experiments.

The optimisation methodology consists of a series of steps for the purpose of performing the design optimisation experiments. The aim is to use different tools and approaches to analyse results and for building robust, versatile, high-performance robots that can utilise available technology to its full potential safely and with fewer design iterations.

4.1 Framework

The optimisation framework is a two-layer scheme, and its overall structure is presented in Figure 4.1. In Layer 1, also called the *Mechanism Layer*, new designs are generated. The designs are found by an optimisation algorithm that seeks the best designs to achieve a set of performance objectives. Each new mechanism is then passed to the *Performance Objective Tester* in Layer 2 (can be also called the Behaviour Layer), where it passes through a series of performance tests. Each performance test can be either: (1) an independent optimisation experiment, or (2) a value that is a result of a calculation. The latter can be an intrinsic property of the mechanism, such as a measure of a physical ability (e.g., how good is a given machine in balancing, which is a measure used in the case study of Part II). The

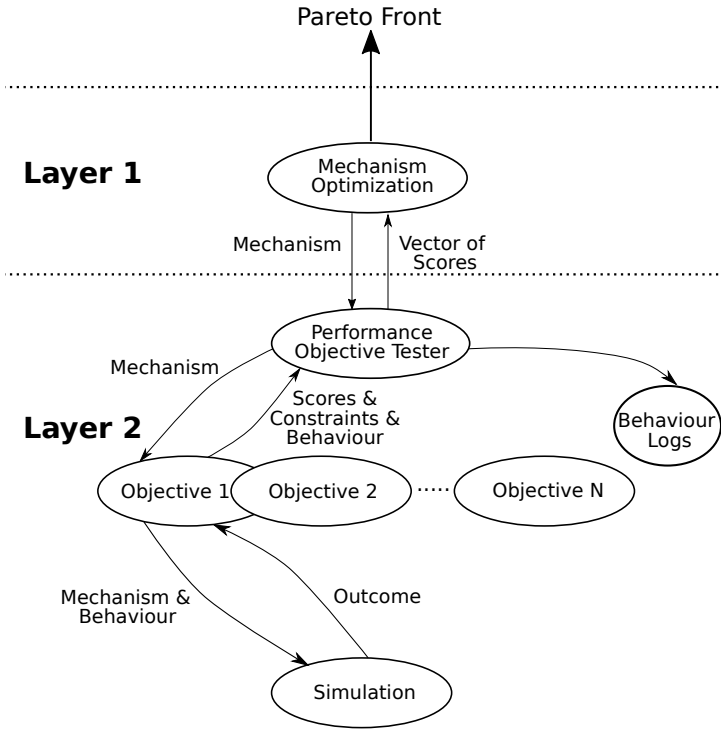

Figure 4.1: Overview of optimisation framework.

former is an optimisation experiment where a given mechanism's optimal behaviour is sought for a given task.

The proposed approach aims to maximise a design's physical ability by meeting or surpassing a given set of performance objectives. For this reason, the focus of this study is on the plant via a behaviour optimisation study, and not the controller of the robot. The framework works under the assumption that if the design is physically capable of achieving the objectives in any feasible way, a controller (or controllers) can be designed to reproduce these behaviours in the real robot.

For each optimisation performance test, the design can fail or succeed, and conditions of failure can be imposed as a set of constraints. The outcomes of each optimisation test of Layer 2 are the following.

1. *Scores*—are values that reflect the performance of the robot for a given objective, they can be either:
   - *a performance score*—a score reflecting its best performance for the given objective, or
   - *an effort score*—any value that can reflect how much effort the system made to reach the given objective.
2. *Constraints*—a vector of constraint values, which are the elements of the equality and inequality constraint expressions.
3. *The optimal behaviour* that achieved the best performance score.

For each performance objective the three results or the intrinsic value (if it is not a behavioural optimisation) are passed on the Mechanism Layer. When all of the experiments in Layer 2 are completed, the optimisation algorithm in Layer 1 evaluates the design's overall performance and generates new designs. As can be seen in Figure 4.1, the framework allows for multiple objectives, which leaves the possibility open for objectives of conflicting nature. The final outcome is a Pareto front of optimal designs and their optimal behaviours for each one of the performance objectives.

Co-optimisation

The presented framework is an example of behaviour and design co-optimisation. This thesis argues that taking into account a detailed description of a robot's performance objectives (the tasks it is designed to perform) in the design process is essential for obtaining more capable, robust and skilful robots. The design and the behaviours of the system have an inextricable relationship which is a topic of interest in various disciplines such as sports science and biology. Dickinson et al. [21] investigated the relationship between locomotion capabilities (behaviour) and structure of various animals, including humans. Animals and humans have different characteristics that promote certain behaviours. For example, cheetahs are faster than humans and in fact they are the fastest land animals on earth. Apes, which are structurally similar to humans are extremely nimble climbers, but humans are not. Even between humans there can be vast differences in performance. In sports, some athletes perform better than others and become champions. Is it hard work, some distinct feature that they have, or both that helps them perform better than others? Optimal behaviours can be obtained by hard work, which means trying repeatedly to improve performance, and the more potential a system has the better the optimal behaviour can be. For that reason the inextricable relationship between a mechanism and its behaviours must not be neglected during the design process.

Performance, Constraints and Effort

An important concept of the framework is the exchange of information between the two layers. Layer 1 searches the design-space for the designs that can achieve the best compromise between all of the performance objectives (a Pareto front). To evaluate each design and efficiently discover the best designs, Layer 1 needs adequate and useful information. Performance objectives could be achieved in many different ways and finding the best way is not easy due to the multiple factors that affect performance. Due to limited resources, the optimiser cannot explore all of the search-space, meaning that it needs useful information for reaching better and faster solutions. For this reason, it is not sufficient for the mechanism layer to know just a succeeded/failed and a performance score, but information on how easy or difficult it was for the design to reach its best behaviour.

The performance score is used to evaluate the fitness of the design for a given objective. The importance of this parameter is evident, since this is what the designer wants the robot to do, and is the main guiding rule during the search process. The effort score reflects how easily the design met a performance objective. It is any value that can be an indicator of how close is the returned

behaviour to the mechanism's true potential (e.g., its optimal behaviour). For example, two different mechanisms might achieve the same objective, but one did it by spending less energy than the other, which means that it required less effort to achieve that objective.

Constraints draw the line between success and failure for a given behaviour. In addition, they also show the margin of success or failure, so that the optimiser can know whether a design easily met all of the constraints, barely met them, marginally failed, or failed by a large margin. For example, if the mechanism easily met the constraints, then there is a chance that it has the potential to perform more difficult tasks. This information is essential for the search process because it guides the optimiser towards optimal designs.

Overview

In the following sections a detailed description on how the proposed approach can be applied for the problem of design optimisation of high-performance mobile robots is presented, together with some of its most important concepts. The approach can be summed in four steps.

1. Building a model.
2. Determining the type of the problem.
3. Selecting Software.
4. Optimisation Methodology.

This chapter discusses several aspects of the design study with the aim to provide sufficient information for tackling the general problem of design optimisation for mobile robots. The steps and techniques presented here are aimed for mobile robots but are not restricted to them.

4.2 Building a Model

In this section the important concepts and steps for building a model of the system are presented. The model includes: (1) a model of the robot's hardware, which completely defines the robot, (2) a model of the robot's behaviour that can capture all the important aspects of the behaviour, and provide enough freedom for the system to explore its true potential, (3) the operational constraints, that set limits on what the system is allowed to do and to what extent, and (4) the performance objectives that define what the robot must achieve.

4.2.1 Realism of the Model

Before diving into details about the steps that are required for building a model, a discussion must be made on the meaning and importance of *realism* in this approach. Realism of a model refers to how close is the theoretical model's behaviour to the behaviour of its real counterpart, and it can

be determined by the level of detail and the validity of assumptions that are made in the modelling process. More accurate models tend to be more complicated, and require more computation time. Less complicated models tend to be easier to understand, require less resources to execute, but might not be as accurate. This creates an important trade-off between simplicity and accuracy of the model, which is discussed in this section.

A realistic design study has many advantages and can lead to:

- fewer design cycles, meaning less time and cost;
- robots that meet their design expectations;
- unprecedented behaviours, and
- efficient use of current technology (reach the mechanism's maximum potential).

Assumptions

Assumptions are an important part of the modelling process. The assumptions made during a study must be clear and thoroughly investigated, for the reason that if they ignore significant information, the results cannot be trusted. Assumptions can also lead to simpler models, more understandable and less computationally expensive.

High-performance robots are fast. This means that dynamics are important, and accurate modelling is necessary so that the real robot can perform similarly in simulation and reality. In this work emphasis is given on the realism of the dynamic models of the robot and its behaviours, which is largely neglected in the majority of the literature, except in a few cases such as Spielberg et al. [104] and Ha et al. [46], that use realistic dynamic models for actuators in their simulations. The aim is to close the gap between simulation and reality so that the desired performance can be achieved without the need of multiple design iterations.

Accuracy Vs Simplicity

Having realistic models seems to have many advantages. However, what are the pitfalls of detailed modelling, and how much effort should be spent when building a mathematical model? An important factor that must be taken into account in the modelling process is the trade-off between accuracy and simplicity. A model should be created after a deep understanding of the underlying mechanism is acquired, followed by validation tests (wherever possible). Even with this though, a perfect model that can take into account every possible variable and scenario is almost impossible to build, and even if we could do that it would require a lot of resources to do so.

Complexity can improve the realism of the model but can result in models that are difficult to understand and can also lead to numerical instability problems (e.g., round-off or truncation errors that lead to loss of precision, crushing or stalling of the simulation). In addition, it comes with the risk

of over-fitting. What if there is a factor we forgot to model or modelled in the wrong way? Then the robot would have been optimised for something different than reality, and will not work very well.

For that reason models must be as simple as possible and in most cases come with/from experimental results for validation. Models will most certainly be imperfect; however, the better they are, the closer the real robot will be to the desired performance requirements. The modelling process described here and the models that are presented in the case study of Part II are created with *simplicity* in mind.

4.2.2 Hardware

The hardware includes all the physical components of the robot and their limitations. Parts with behaviours that have an impact in the overall performance of the robot must be thoroughly modelled (e.g., the motors), while other parts with insignificant behaviours can simply have their masses added in the model (e.g., dust covers protect components of the robot from getting contaminated by dust, but for a design study simply including their mass in the model would suffice).

In this part, a general discussion on important components of robots and concepts for generating numerical models for the purpose of simulation and optimisation is presented. Building such models requires a deep understanding of how robots work as well as the requirements of the system for the desired applications. Examples of such models could be the following:

1. a dynamic model of the mechanism, including a kinematic model and limitations;
2. models of sensors, including limitations (e.g., noise, drift, saturation, resolution, time delays);
3. models of actuators, including limitations, and
4. models of other parts, such as transmission mechanisms, power supplies, joint motion end stops, springs, dampers, crash protection, and limitations such as: voltage, current, speed, force/torque, stroke, hysteresis, friction, efficiency, stress/force limits on parts (especially bearings).

Not all of the aforementioned are necessary for all robots and applications. For example in the design optimisation of a soft robot the topology and the properties of the material that make up its body are essential for the study because they directly affect its behaviour (e.g., Hiller et al. [50] and Cheney et al. [17]), but such properties are not needed to model a robot made of stiff (almost rigid) parts.

Dynamic Model

A dynamic model is a set of data from which the robot's equation of motion can be calculated. The dynamic model includes:

- a kinematic model and its limitations (e.g., motion range limits for avoiding self collision and singularities);

- mass and inertia data of each body;
- material properties and limitations, which are relevant for soft robots such as: stiffness, stress/force limits, or density.

Kinematic Model—defines an abstract representation of the robot's skeleton sufficient to describe the robot's degrees of gross motion freedom, and can include open-loop and closed-loop kinematic chains (e.g., a four-bar linkage). In more detail, the kinematic model defines:

- the number of bodies (or links) and joints (which are the same in the absence of kinematic loops);
- *connectivity data*: define how and in which order the bodies are connected together;
- *joint data*: define the type of joint with relevant joint parameters, and
- *geometry data*: define where the joints are located in each body.

Connectivity data are relationships describing how bodies are connected to each other. As an example, in special cases *kinematic trees* can be used to describe connectivity in rigid body systems (Featherstone [29]).

Joint Data may include special joint parameters (if there are any) and the types of joints, which for a rigid robot with the most general ones being the following.

1. *Prismatic joints*: the relative motion between bodies is a translational sliding motion parallel to a straight line.
2. *Revolute joints*: the relative motion is a pure rotation about an axis that is fixed in both bodies.

Geometry data defines the location of each joint relative to two body coordinate frames.

Mechanical Limitations arise from material properties and the geometry of parts. Modelling them can lead to more robust designs and behaviours. By performing a crude dynamic analysis on the various parts of the robot an estimate of the forces on the joints and parts can be calculated. The analysis can be performed on the most demanding behaviour of the system. These estimates can define an upper limit on the forces that the part/system will be under. By using these limits, appropriate mechanical components can be selected and their limits can serve as force/torque constraints which must not be violated during the behaviour optimisation. In this way the design will co-evolve with behaviours that do not surpass the stress limits of parts, and produce a more robust system.

An example of such components are ball-bearings, which come with a maximum load that they can withstand before breaking or deforming. Stronger bearings can be bought, but they are bigger and heavier, which will result in bulkier and slower robots. For this reason, it is wise to set an upper limit

to forces based on a preliminary study to ensure that appropriate parts for designing the robot exist.

Rigid Robots—assuming rigidity when designing or controlling a robot is a common practice that leads to simplified and easier to understand models. Under this assumption the links of the robot (except explicitly stated such as in the case of springs) do not undergo deformation. In reality all bodies under a load experience some form of deformation; however, many robot links are built with stiff materials (such as aluminium alloys or fibre composites) and this simplification of the model results in small model errors. For rigid body systems there is a variety of algorithms for estimating the states of the links (such as the Recursive Newton-Euler Algorithm (RNEA) for inverse dynamics and the Articulated Body Algorithm (ABA) for forward dynamics [29]).

Soft robots in contrast, are robots containing compliant bodies. These robots undergo significant deformation, and the assumption of rigidity can no longer be used. In soft robots kinematic and dynamic modelling is more challenging than rigid robots. Scientists approach the modelling problem with simplifying assumptions (such as the *Piecewise Constant Curvature* (PCC) model in Webster and Jones [109]) or finite element based modelling (Rus et al. [90]). The first method leads to some loss of accuracy due to the assumptions taken in the examined model, while the later leads to intense computations that require a lot of time to complete. The framework proposed in this thesis is applied to a specific case of a rigid-body robot; however, as is explained in Section 4.6.4 the framework could be applied to any type of robot (including soft robots) with a well defined dynamic model, behaviours and objectives.

Models of Sensors

To adapt in dynamic environments and produce robust behaviours, robots need to perceive their surroundings through sensors. Sensing is an integral part of the control stage of a robot. So why is it not also a part of the design process? A system can react to external stimuli only if it can perceive them. If the system cannot sense a stimulus, perceives it with a delay, or even without adequate information, then it cannot react properly to it, which leads to sub-optimal behaviours. To achieve top physical performance the robot must know what are the limits of its sensorimotor capabilities and adapt its behaviours to them. For this reason, models of sensors (and actuators which are explained in the next section) and their limitations must be taken into consideration in the design process.

Sensors are input devices which produce an output signal to quantify a specific physical quantity (i.e., the input). There exists a variety of sensors for measuring different quantities. The types of sensors a robot might need to be equipped with varies according to the application. Important types of sensors are *state estimation sensors*, which provide information about the robot's states such as its position, velocity, orientation etc. Examples of sensors are shaft encoders, tactile sensors, temperature sensors, accelerometers, LIDARs or Inertial Measurement Units, which use a combination of different sensors

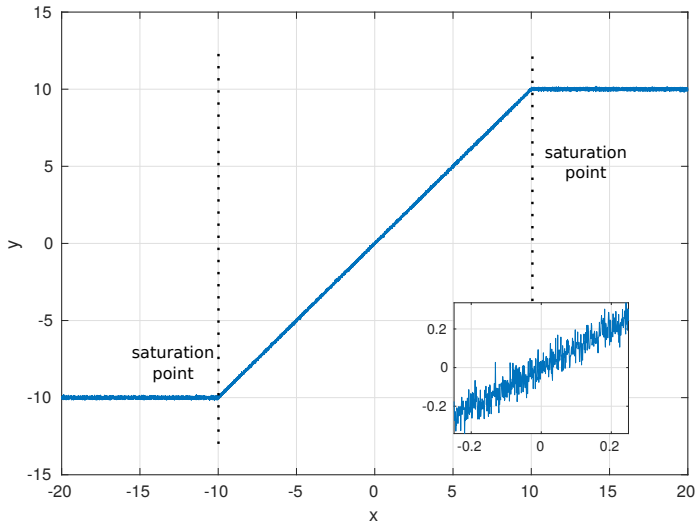

Figure 4.2: Example of saturation behaviour when an IMU's accelerometer maximum range is reached, in a sensor with an equal positive and negative maximum range. x axis values represent time and y axis values are g. Inner figure is a zoomed area demonstrating an example of Gaussian noise with zero mean and Root mean square (RMS) of 0.05 g.

to measure orientation, angular rate and accelerations. All sensors inherently come with limitations which may be, but are not limited to the following.

- **Measurement noise**—sensor signals are polluted with noise. The source of this noise can be electrical or mechanical and depends on the environment of the sensor (e.g., factories are noisy environments). The most common type of noise in sensors is Gaussian, and a parameter for quantifying sensor noise that is commonly provided by manufacturers in the sensor data-sheet is *Noise Density*. An example would be to use this parameter to build a model that can be incorporated in the simulations to examine how the system and its controller evolve with realistic uncertainties in the system's measurements. An example of such experiment is performed in Featherstone [31], and a visual representation can be observed in Figure 4.2.

- **Range**—the range of the sensor is the maximum and minimum values that can be measured. Depending on the application this limitation may also have a great impact on the design and its behaviours. Not all robots need to operate near the limit of their sensing capabilities, but these limitations should be taken into account during the design study. In average cases they provide an upper limit on the robot's operational range and in extreme cases they can even directly affect a behaviour which can be optimised to avoid or operate near these limits. For example, high-performance mobile robots might experience extreme accelerations in a case of crush landing or even during normal operation. If these accelerations surpass the limits of an equipped accelerometer (e.g., saturation limit), the sensor might be damaged or not provide reliable measurements for some time. Behaviours that take into account such limitations are

examined in the case-study presented in Part II, and a visual example of a saturation behaviour can be observed in Figure 4.2.

- **Time delays**—sensors do not instantly change their output when a change in the sensed quantity occurs. Additional delays can be added by processing of the data (e.g., filtering, extra computations etc.) causing the system to perceive its environment with a delay. Depending on the application this may have an impact on the controller of the system, and might be worthy to examine, especially in fast robots.

Before proceeding with the modelling part a research on available sensing technology and an evaluation of available resources for obtaining it must be performed. The designer must examine the available sensing technology to determine which sensors are needed for the examined system, and selection must be made according to the costs of the sensors, their availability and other project specific requirements. These requirements may include sensor accuracy, power consumption, sensitivity, precision, resolution, offset errors, linearity, hysteresis, quantization errors (in digital sensors) and more. An elaborate discussion about the different types of sensors, their applications and their limitations can be found in Fraden [35].

Depending on the application and the optimisation process different limitations can be incorporated into the study. In the case of an open-loop control (i.e., behaviour optimisation) feedback related parameters such as measurement noise are not relevant. In the case of controller optimisation, feedback-related parameters such as time delays and measurement noise can be incorporated into the model for exploring robust designs and controllers. Even if the sensor limitation values are not exact (e.g., assuming a Gaussian sensor-noise distribution with a given mean and standard deviation), integrating them to the simulations can help study the effects of such limitations in co-evolution of the design and its behaviours, and result in more robust robot designs.

Further analysis that not necessarily need to be included in the optimisation process may include redundancy of the sensors and their distribution on the robot. For example, if a controller is optimised that requires as input an angular acceleration value, the robot needs to be equipped with at least two IMUs. In systems that experience strong shocks due to rapid body accelerations/decelerations (e.g., a legged robot might experience this during landing after a high jump, something that might saturate its accelerometers), careful consideration must be taken during placement of the sensors. Singh and Featherstone [102] propose a method that uses the Centre of Percussion (CoP) for placing sensors on legged robots, in such a way that the effect of shocks on the sensors during locomotion is minimal.

Models of Actuators

In the previous section the need of sensing models was discussed. Sensing and actuation are two different but inseparable items and are usually referred to as one instance with the term *sensorimotor* capabilities. Sensors perceive the environment and estimate the robot's state, while actuators are

responsible for generating motion. Actuators require an energy source and a control signal to convert the former into mechanical motion.

Actuator models play a crucial role in achieving the required performance, and are responsible for producing the power that the system needs to move and subsequently reach its goals. They come in various forms and can be coupled with gears for producing more torque or speed.

The most common types of actuators in mobile robots are the following.

- Hydraulic: produces power from pressurised fluid (usually oil).
  - Pros: suited for high-force applications.
  - Cons: bulky, noisy and require regular maintenance, can leak fluid and lose efficiency or damage surroundings.
  - Power source: a compressor.
  - Control signal: an electrical signal sent to the hydraulic valve.
- Pneumatic: produces power from pressurised air.
  - Pros: operation in extreme conditions, are naturally springy, and lightweight.
  - Cons: pressure losses can lead to less efficiency, requires maintenance, they are not very precise and they are noisy, and might require some kind of gear to convert the high-speed low-torque output of the motor to something with lower speed and higher torque/force.
  - Power source: a compressor.
  - Control signal: an electrical signal sent to the pneumatic valve.
- Electric: produces power via interaction between an electric current and a magnetic field.
  - Pros: higher precision and higher bandwidth than hydraulic and pneumatic actuators, quiet, compact and small.
  - Cons: can overheat, they are not suited for all environments and can be more expensive than hydraulic and pneumatic actuators.
  - Power source: electric current.
  - Control signal: a voltage sent to the motor driver circuit.

Including the actuator's dynamic model and optimising its input signal results in a more realistic optimisation study. Moreover, in contrast to optimising an output value (such as forces or torques), taking into account the actuator's dynamic model and its limitations results in behaviours that have a higher chance of being reproducible and feasible in reality (e.g., not all torque profiles can be achieved by all actuators). Limitations of actuators can be force/torque, speed, power, thermal limits and more.

Off-the-shelf actuators usually come with a set of specifications that are detailed enough to build a realistic model of the actuator. An example of such a dynamic model is presented in Part II, where a

Figure 4.3: Left: Hyq2Max [98] demonstrates the power of hydraulic actuators by pulling a three-ton plane; right: a high-performance balancing robot equipped with small and fast brushed DC motors (see Section 5.5).

model of a DC brushed electric motor is built based on supplier specifications.

Actuator Selection—actuators should be selected based on the intended applications of the robot; however, their selection can also be a part of the optimisation process. They can be modelled in the optimisation as discrete components to form a mixed-integer optimisation problem. Nevertheless, an initial analysis on the performance objectives and the robot can provide enough information for selecting an appropriate actuator, and if resources allow, different actuators can be part of the optimisation study. For example, if the robot is designed for highly dynamic behaviours an actuator with high-power output should be selected, while a robot that is designed to lift or displace heavy objects needs actuators with a high-torque output, such as hydraulic actuators.

Models of Other Parts

In this stage remaining parts of the robot's mechanism that effect its performance should be modelled. These are parts that either have a direct affect on the behaviour of the robot (e.g., a transmission mechanism changes the behaviour for the actuation system), or have mechanical limitations that must not be surpassed (e.g., the maximum force a bearing can withstand). Examples of such parts are the following.

- **Power supplies and their limitations**—for example an electrical motor and the battery have an upper limit on the amount of current or voltage that can be drawn/supplied.

- **Transmissions** are mechanisms for transmitting and transforming power. They can be gear-boxes, clutches and direct drive mechanisms. These mechanisms can have losses due to friction, speed limits and limitations on the maximum amount of force/torque that they can withstand.

- **Energy dissipation components** for improving the robustness of the robot, examples of which, are the following.
  - End-stops—are soft springy parts for dissipating energy. They help prevent the robot from harming itself from self collisions. Most robots should have them for preventing self damage. In special cases they might also be actively used in a behaviour (e.g., dissipate energy when the robot needs to stop).
  - Crash-protection—is a component for improving the robustness of the robot. In real experiments several things can go wrong (including software errors) and undesired scenarios will occur, such as a crash landing. All the previous parts were useful during normal operation, but this part exists solely to minimise the damage in undesired scenarios.
- **Energy storage components** such as springs, and their limitations, which are used for storing and recycling mechanical energy. Depending on the application they can be made of different materials and in different shapes. In highly dynamic motions these components are responsible for recycling the necessary energy for achieving performance objectives, and hence are very important. Their profiles and limitations such as hysteresis, yielding point, and efficiency can be evaluated via stress-strain tests (see Section 5.3.1).

Physical Constraints

Because the mechanical model already includes many *physical constraints*, it is reasonable to treat modelling and physical constraints together. As mentioned in the introduction of this chapter, the constraints define the line and the margin between success and failure for a behaviour. They can be equality or inequality constraints, and could be divided into two main categories.

- Kinematic Constraints: e.g., avoid self-collision, joint limits etc.
- Mechanical Constraints: e.g., actuator torque limits, sensor saturation limits, yield points of parts and more.

Scaling—the constraints must be converted to equality or inequality constraints to be used by the optimisation algorithm. However, the constraint values might be of different orders of magnitude, which might mislead the optimiser. For this reason it is advisable to scale the constraints.

For example, ground-force reactions can be in the order of hundreds of newtons, but joint limits can be a few radians. A constraint violation of 1 N (which is very small) is not the same as a joint rotating 1 radian past its limit. The same reasoning can be applied to the objective function values.

4.2.3 Behaviour

A behaviour is the actions that an individual system, or in the context of this thesis a robot takes. These actions are performed for achieving a performance objective. In the proposed optimisation approach, a robot's behaviour to achieve a set of performance objectives is optimised.

The main idea behind the proposed approach is the maximisation of a robot's physical performance. This means that the plant is the main focus of the presented work. Modelling of a behaviour can vary for different examined cases. In this thesis the following general categorisation of the behaviour parameters is proposed.

1. Initial parameters,
2. outcome parameters, and
3. input parameters.

Initial Parameters

Initial parameters describe the state of the robot, and other relevant quantities at the beginning of the behaviour. They can be independent or dependent variables. Examples of initial parameters are state variables, such as positions (e.g., of joints), velocities, temperature and other key variables such as accelerations. These parameters, together with a set of assumptions, should be sufficient to describe all the quantities of interest at the beginning of the simulation.

Input Parameters

These parameters describe the actions that the robot takes during the simulation. They are independent variables and are decided by the optimisation algorithm. They can be: (1) an open-loop behaviour such as applied forces/torques or the input to the actuator (e.g., voltage, current or pressure), or (2) a closed-loop behaviour such as the gains of a parametric controller. Open-loop behaviours evolve over a time window, which means that time duration parameters must also be defined with them. The values that these parameters are allowed to take are bounded, and their limits are defined by the limitations of the related components (e.g., power supply and actuator), or other behaviour time limitations. For example, in the case of an electric motor these values cannot surpass the maximum voltage that the batteries can supply, or the maximum current of the motor.

Outcome Parameters

Input and initial parameters are actions taken by the robot, while the outcome parameters are dependent parameters that describe what those actions achieve. They are things that must be true either in the duration of the entire behaviour, a part of it or the end of it. Examples of these can be: forces that must not surpass a given limit, a desired final condition (e.g., have a given CoM velocity at the end of the behaviour), a percentage of the behaviour duration that an event occurred (e.g., motor current saturation for 45% of the behaviour).

Behaviour or Task Constraints

Behaviour or task constraints define the way a behaviour must be executed. They can be:

- things that the robot should do or achieve (during the entire period of the behaviour or just at some point of it e.g., be at a specific state when the behaviour ends), and

- things that the robot shouldn't do during a behaviour (e.g., the behaviour shouldn't last more than 0.3 s or the robot shouldn't excessively slip during a behaviour).

Scaling—similarly to physical constraints, behaviour constraints might have different units and their values might be in different orders of magnitude, hence to be effectively used by the optimiser they must be scaled.

4.2.4 Performance Objectives

Performance objectives quantify what we want the robot to achieve. This section presents a discussion on how the performance objectives can be selected, the concepts behind their selection, and how they can be mapped into optimisation objectives. The guiding principles behind the choices are: versatility (if required), reachability, approaching the performance envelope, and minimality.

Performance Envelope

Before embarking on the design study the performance envelope must be defined. The performance envelope refers to the capabilities of a design in terms of the desired task (such as speed and power). After a general idea of what the robot needs to achieve is defined, the most demanding performance objective of the robot should be sought.

Determining the performance envelope enables the designer to make initial design decisions (such as how many springs the robot will have, what type of motors will be used and more). These results can be obtained via simple calculations or via preliminary optimisation experiments (e.g., if we want a robot to perform very high vertical jumps we can simulate only this behaviour to determine which is the highest jump it can achieve).

Versatility

In industrial environments, conditions are controlled, and hence mostly predictable. Industrial robots, which are usually stationary, need to perform one task or a family of closely related tasks such as assembly/disassembly of parts. However, in real world scenarios, mobile robots need to cope with high uncertainty and demonstrate multiple different behaviours. In addition, these behaviours can place conflicting requirements on the robots, making design decisions even harder.

As mentioned in Section 2.1, multi-objective problems represent a class of problems that are generally difficult to solve, and require sophisticated software. Furthermore, with more behavioural objectives come more simulations.

Minimality

This concept simply means keeping the number of objectives as small as possible in order to keep down the required resources of the design study. Every behavioural objective is an optimisation experiment by itself. This means that each new performance objective added may result in a substantial computation time added.

Reachability

The term 'realism' does not only apply to the robot's models, but also to how the robot achieves its objectives. Reachability refers to the fact that many performance objectives have prerequisites. The behaviours must be achieved in a similar manner that they would have been achieved in the real world. Behaviours must be a chain of interconnected steps that the robot can achieve, starting from a home configuration and be able to return to that point in some way.

As mentioned in the introduction, performance objectives can be either: (1) an inherent property of the robot, or (2) a behavioural objective. One example of the former is to minimise mass or size, and for the latter is to find a behaviour that can reach a given goal such as to perform a 2 m vertical hop. For example, in the case study presented in Part II, a monopod robot is optimised to reach its highest vertical hopping height. However, the robot can achieve its highest hop only after a fall from a given height. This happens because the robot needs to recycle stored energy in its springs. This means that for achieving the latter objective, a series of intermediate performance objectives (which are increasing height hops) must be achieved.

Moreover, a study is not complete if only the behaviours for reaching an objective are defined. Behaviours for returning to a home state must also be studied, unless doing so is trivial. Coming back to the case study of Part II, when making a 2 m vertical hop we also need to define what will happen afterwards. If the robot ends up crashing after reaching that height, then this robot will have zero practical use.

Optimality of Sequence

In an optimisation study, a robot and its corresponding optimal behaviours for achieving a specific set of objectives can be sought. In this case the robot must achieve an objective with specific initial and final conditions. In other words, we want the robot to meet an objective by a predetermined series of intermediate objectives. This simplification reduces drastically the search space and produces faster results; however, it does not allow the optimiser to search for optimal ways for reaching its performance envelope.

A more general problem would be to allow the optimiser to perform a series of intermediate experiments to find the optimal way of reaching its performance envelope. However, this problem is at least exponentially more difficult to solve than the problem with predetermined intermediate objectives. In the case-study of Part II, a monopod robot is optimised to reach its maximum vertical

hopping height. To do so, it is required to perform a series of increasing-height hops. This can be achieved by allowing the optimiser to explore a connected sequence of hops, beginning and ending at rest, such that the first few build up to a maximum-height hop and the last few come back down again.

Mapping Performance to Optimisation Objectives

To be used by the optimiser, performance objectives must be mapped to optimisation objectives. The latter, can be any of the following.

1. Reaching a given performance goal, in this case we are minimising an error term, such as: $\min \left|\left| x_{\text{outcome}} - x_{\text{target}} \right|\right|_2$, which defines a quadratic objective function which has generally desirable properties (Section 2.1).

2. Seeking the best value for a given performance objective. Could be either a maximisation or a minimisation objective.

4.3 Determining the Problem Type

Having obtained a model, a set of optimisation objectives and a set of constraints, the problem type can now be determined. The importance of this step was extensively discussed in Section 2.1. The designer needs to identify what is known about the nature of the formulated design optimisation problem so he can: (1) understand the complexity of the given problem (how difficult it is to solve), and (2) select the appropriate optimisation techniques for solving it.

As presented in Chapter 2.1, any given optimisation problem can be classified in any or many of the following categories.

1. Continuous Vs Discrete Vs Mixed Integer Optimisation.
    - Continuous: all the independent variables are continuous.
    - Discrete: all the independent variables are discrete (e.g., natural numbers).
    - MI: some independent variables are discrete and some continuous.
2. Unconstrained versus Constrained Optimisation.
3. None, One or Multi Objective Optimisation.
4. Convex Vs Non-Convex Optimisation.
    - Convex: the objectives are convex and the feasible set is a convex set.
    - Non-convex: if convexity can't be determined, then the problem can be considered to be non-convex.
5. Non-linear Vs Linear Optimisation.

- Linear: the objectives and constraints are linear (the problem is also convex).
- Non-linear: at least one objective or constraint is non linear.

6. Deterministic Optimisation Vs Optimisation Under Uncertainty.

   - If the model contains random elements with a known distribution (e.g., any of the objective functions can produce a different output for the same input), then it is stochastic.

In complicated models it is sometimes difficult to tell if the problem belongs in some of the aforementioned categories. For example, problem types 1, 2 and 3 are part of the problem's definition so the designer can decide if the problem can be formulated to belong to any of these categories. In problem type 6, the designer can also decide if uncertainty should be included in the model.

However, problem types 4 and 5 depend on the mathematical properties of the objective function and the constraints. Determining if a system is not linear can be easily verified by checking either the mathematical formula of the objective function or by sampling the model. The big challenge lies in convexity. If convexity can be verified, then a series of efficient algorithms can be applied for solving the problem (Section 2.2). Proving convexity however is not an easy task if a closed form solution of the model does not exist. For this reason, it makes sense to treat problems where convexity cannot be proved as non-convex, and solve them using global optimisation techniques. In any case, local optimisation techniques can also be applied, but only one solution will be returned.

4.4 Selecting Software

The next step in the optimisation process is to select an appropriate software for the type of the examined optimisation problem. Optimisation software comes in three kinds of packages:

- *Modelling software* to formulate the problem.
- *Solver software* for running the simulations.
- *Optimisation software* with optimisation algorithms and analysis tools to address the examined type of problem.

4.4.1 Modelling

Modelling software is used for formulating the problem on a computer. In this software the model of the mechanism and the behaviours can be interpreted in a software language or workflow for analysis. Various modelling software platforms represent models with different approaches (such as a workflow or simply by coding), integrate with other applications, or differ in the ways they process results and invoke solvers. Despite that, all modelling software has one purpose, and that is to produce computational models to be executed in a computer.

The modelling software must be simple enough to use and provide enough freedom for modelling and simulating all the important aspects of the robot's mechanism and its behaviours. The input to the software are all of the variables (both dependent and independent) that define the models, and its outputs are values for visualisation, producing constraint and objective value functions, or for any other purpose. There is a large list of available simulation software for every need such as Simulink, Adams, ANSYS and more.

4.4.2 Solver

Solver software are used for solving mathematical problems via numerical analysis, and they usually come together with the modelling software. They are concerned with finding solutions to specific instances of the examined problem, and in this thesis context to simulate the behaviours of mechanisms.

Selecting a solver is an important step because it plays a major role in the: (1) quality of solutions (they introduce truncation errors), and (2) the computation time of each problem instance evaluation. Solvers use different methods for solving the numerical integration problem that can be categorised according to their *order*. Methods with higher order are more accurate, but require more time to execute. Depending on the problem, the type and the optimisation method an appropriate solver can be selected. If a gradient-based optimisation algorithm is used for solving the problem there is a need for numerically accurate simulations, and hence a solver of high order must be selected for exploiting efficiently the gradient information. More details about simulation and modelling can be found in Law [60].

4.4.3 Optimisation

Upon acquiring a computational model of the examined system, an appropriate optimisations software must be decided. Based on the problem type (Section 4.3) an appropriate numerical optimisation method must be selected. Just like the modelling software, there is a variety of commercially and open source available optimisers.

Each software comes with its own implementation of optimisation algorithms and various tools that assist the optimisation process. The factors that must be taken into account for selecting an optimisation software are the following.

1. The optimisation software must have a *bridge library* with the modelling software so they can exchange information (Figure 4.1).

2. The software should have an implementation of at least one algorithm that is appropriate for the examined problem type.

3. The software is efficient, meaning that it has efficient implementations of optimisation algorithms, fast communication between all the interconnected software, and is easy to use.

4. It is desirable if the software also has pre-processing, post-processing and data visualisation techniques, if not an additional software for this reason must be selected. Interpreting and analysing the data is a very important step for the overall optimisation process (see Section 4.5.5).

4.4.4 Proprietary Vs Open Source

At the time this thesis is written a large variety of proprietary and open source platforms exist for the previously mentioned software. Selecting a software depends on various factors such as availability (if the designer already has the software), familiarity (if the designer already knows how to use it), simplicity, appropriateness (if the software contains the desired optimisation method), and cost of the licence.

Open source software as well as proprietary software are widely used in industry and academia under reasonable conditions (such as Matlab's Optimisation Toolbox [68], Gurobi or ModeFrontier [27]). However, a large portion of software for global optimisation is proprietary, which is usually more robust and more reliable, but comes with a high price.

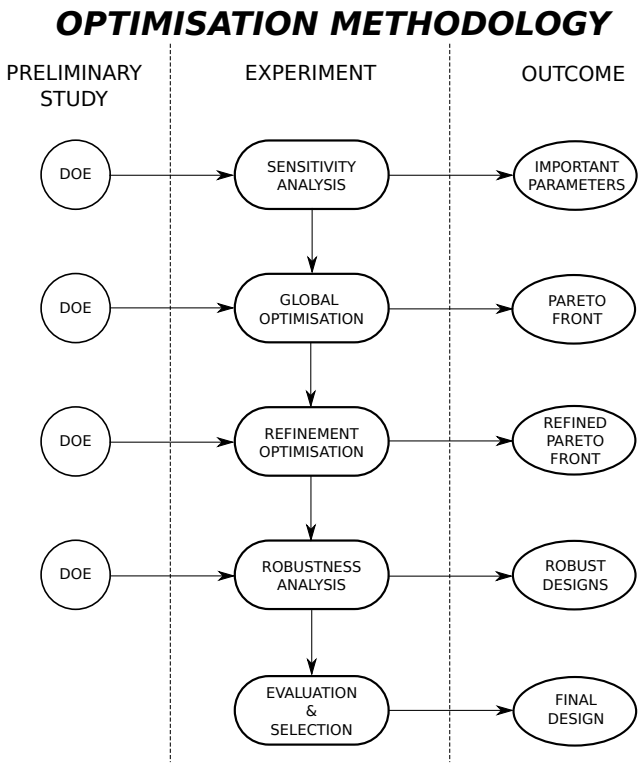

Figure 4.4: Optimisation methodology overview. Prior to each experiment a DOE is performed to obtain qualitative initial information. Each experiment uses the optimisation framework presented in Section 4.1.

4.5 Methodology

The previous sections described how to set-up the problem. In this section a methodology for running the optimisation experiment is proposed. The optimisation methodology can be described by six steps, with each step having a specific purpose and reasoning. These steps help improve the quality of the obtained solutions and allow the designer to use appropriate criteria and tools for selecting the best design depending on the application. The overview of the proposed optimisation methodology is presented in Figure 4.4. In summary, the proposed steps are the following.

1. *Preliminary design space exploration* is performed with DOE techniques to maximise the quality of initial information using a small sample of the search-space. This step is performed independently and prior to each experiment.

2. *Sensitivity analysis* for gaining a deeper understanding of the examined problem, for making early design decisions, and for selecting the most important parameters to optimise.

3. *Rough optimisation* for finding approximate local optima by using global search algorithms.

4. *Refinement of the Pareto optimal set* by using local optimisation algorithms to improve the quality of the Pareto front.

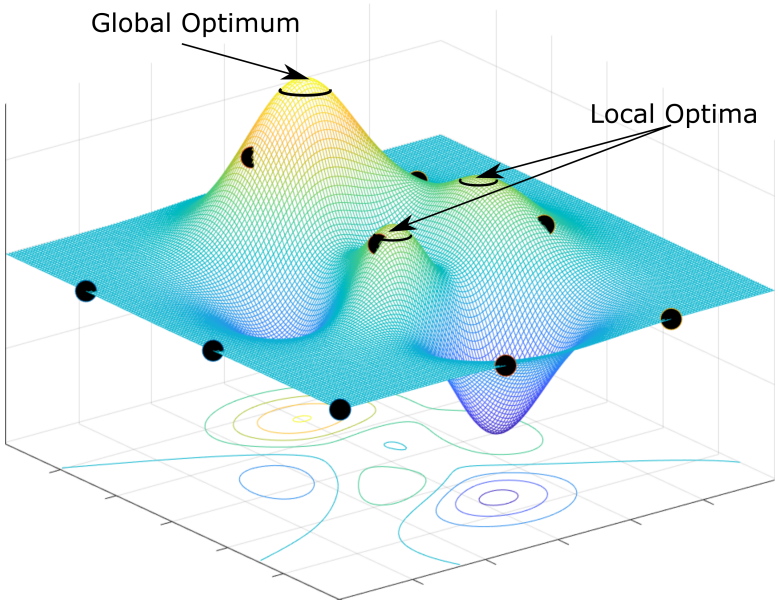

Figure 4.5: DOE for global optimisation example in an arbitrary 3 D function. Black spheres represent initial points for the optimisation process. Using an appropriate DOE method can result in an efficient coverage of the search space, with some initial points ending up close to some of the objective function's optima.

5. *Robustness analysis* for finding the most robust designs in the presence of expected uncertainties in the real systems (e.g., manufacturing errors, initial conditions etc.).

6. *Post-optimisation analysis* for evaluating the results based on a set of criteria.

In the following sections, a detailed explanation of each step, its purpose, its drawbacks and its advantages is presented.

4.5.1 Design of Experiments

An important preliminary step before any optimisation experiment is to maximise the quality and quantity of initial information. Qualitative information can lead to: (1) reduced time and resources for the experiment (i.e., fewer experiments, faster convergence of algorithms, and higher completeness), (2) better coverage of the search space, (3) improved understanding of the examined system, and (4) higher chances of finding a global optimum. This is performed using Design of Experiments (DOE) techniques. DOE are techniques for generating initial samples of the design space, which can serve for various experiments and purposes.

Applications

DOE techniques can be used for various applications. In the context of robot design, the following DOE categories are of interest:

1. DOE for statistical analysis,
2. DOE for optimisation, and
3. DOE for robustness analysis.

DOE for statistical analysis is used for extracting the most relevant qualitative information from a small set of samples. These techniques are suitable for sensitivity analysis. They sample the search-space in a way that a sufficient coverage is achieved and redundant information is eliminated (they produce low-correlation samples). This means that the generated values are distributed in a wide range of the search-space. Examples of such algorithms are *factorial algorithms* (Figure 4.6), *Latin Square algorithms* and *Monte Carlo* sampling methods (Saltelli et al. [94]).

DOE for optimisation is used for generating suitable initial populations for optimisation experiments. By having a favourable distribution of initial samples over the search-space the chances of landing close to the global optimum increase. Having said that, robust algorithms (Section 2.2.1) should not be affected by the quality of the initial population, but only for a sufficiently large number of iterations, which available resources do not always allow. Regardless of that, proper initial population sets increase the chances of a faster convergence to the global optimum, which is always preferable.

A common approach in global optimisation algorithms is to have a portion of the initial seeds generated by a DOE for exploration, and a portion consisting of seeds that are known to perform well for exploitation. Suitable algorithms for DOE optimisation are *Space filler algorithms*. These algorithms incrementally generate new designs by maximising the distance (in the search-space) between the new and the previous samples, or by uniformly sampling the search space (e.g., Monte-Carlo algorithms).

DOE for robustness analysis aim to find designs such that their objective function values lie on flat peaks. This means that small variations of the input variables will not significantly deteriorate the robot's performance. This method increases the chances that the real design will perform well even in the presence of uncertainties and imperfection in the real world. These techniques are used to sample points with specific mean and variance around a given point (e.g., Monte-Carlo methods). Details about robustness analysis are presented in Section 4.5.6.

4.5.2 Sensitivity Analysis

The aim of this step is to determine which of the design or behaviour parameters (or combinations of them) have the highest effect on the performance of the robot. By determining the parameters that the system is most sensitive to, the designer can achieve the following.

- Reduce the number of independent variables. Parameters that have small effect on the performance of the robot can be fixed at a specific value. This can reduce the computational resources required by the experiment.

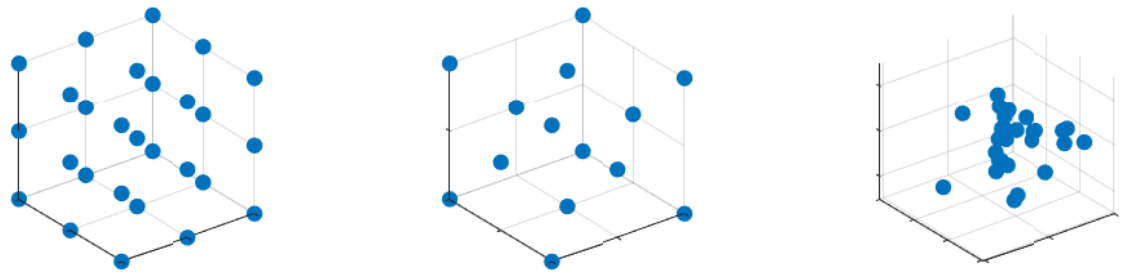

Figure 4.6: Left: Full Factorial algorithm which examines every possible combination of variables in the search-space, but is computationally inefficient for a large number of inputs. Middle: Reduced Factorial algorithm provides a reasonable, low correlated and efficient coverage of the search-space. Right: randomly distributed points that have a low coverage of the input-space.

- Gain a deeper understanding of the examined system, which can aid design decisions.
- Identify the most sensitive parameters in the performance of the robot. It is important to get these parameters right in the real robot. For example if the design is sensitive to variations of 1 mm in a design parameter, then the corresponding component must be manufactured with an error tolerance below 1 mm, so the design's performance does not deteriorate. In addition, components with low sensitivity can have less strict tolerances. This knowledge is important because manufacturing cost rises with stricter tolerances.

This procedure requires an initial population to be well distributed in the search-space and to have low-correlated input variables. DOE for statistical analysis techniques are suited for generating an initial population set (Section 4.5.1). Some examples of initial sampling of the input space for sensitivity analysis are presented in Figure 4.6.

Sensitivity analysis can be performed with various tools and methods, and the variables of interest are the following.

- *Factors*—are the independent or input variables.
- *Responses*—are the output or the objective function values.
- *Main effects*—are this step's output, and quantify the effect that a single factor has on the responses.
- *Interaction effects*—if the effect of one factor changes based on the effect of another factor then that is an interaction effect. Interactions between two or more variables can be examined.

The main idea behind sensitivity analysis is to apply changes on an input variable while keeping the other variables constant, and measure the percentage of change in the objective function. Sensitivity analysis can be performed with statistical methods that estimate the effect of each factor to the global variance. The effect can be estimated using a variety of methods such as regression methods (Saltelli et al. [94]).

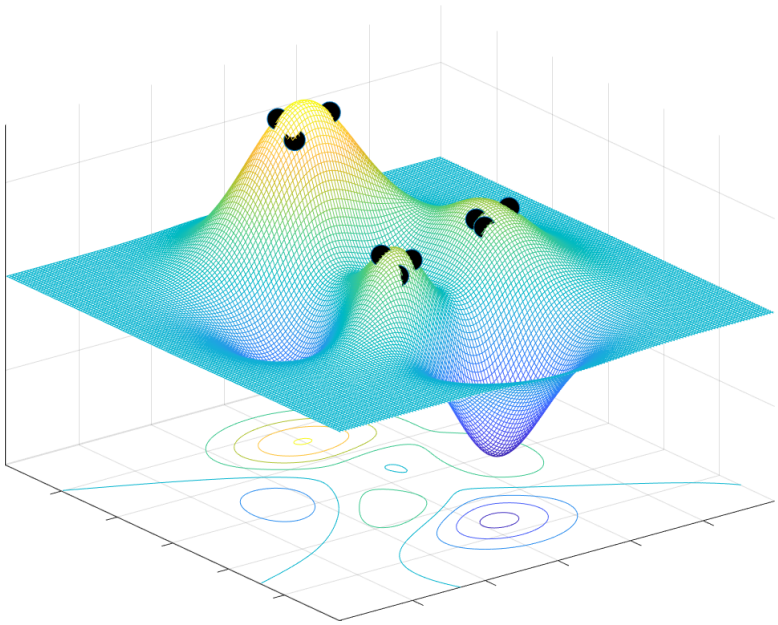

Figure 4.7: Example rough optimisation of the function presented in Figure 4.5. Rough optimisation algorithms do not have high accuracy. For this reason, the final result could be solutions in close proximity to the true optima.

4.5.3 Rough optimisation

In this stage global optimisation techniques are employed for finding *rough* or *approximate solutions*. As mentioned in Section 2.2.3, global optimisation algorithms use metaheuristics that are mostly stochastic for generating fast solutions that are *close enough* to optima. These methods are less sensitive to modelling errors, and escape local minima to eventually reach the global optimum. This step can be described by the following process.

1. Generate an initial population set with DOE optimisation techniques. If already known solutions exist (from preliminary experiments or intuition) that provide reasonable results, they can be added to the initial population set.

2. Select a global optimisation method (see Section 2.2.3).

3. Define the number of algorithm iterations and trade-off between exploration and exploitation by selecting an appropriate implementation of the algorithm, and by tuning the hyper-parameters of the algorithm (note: tuning of the hyper-parameters requires a deep understanding of the examined problem and the optimisation algorithm).

4. Run the optimisation experiment.

These algorithms require an initial population set that can be generated from the DOE techniques described in Section 4.5.1. With a sufficiently well distributed and large initial population set, an

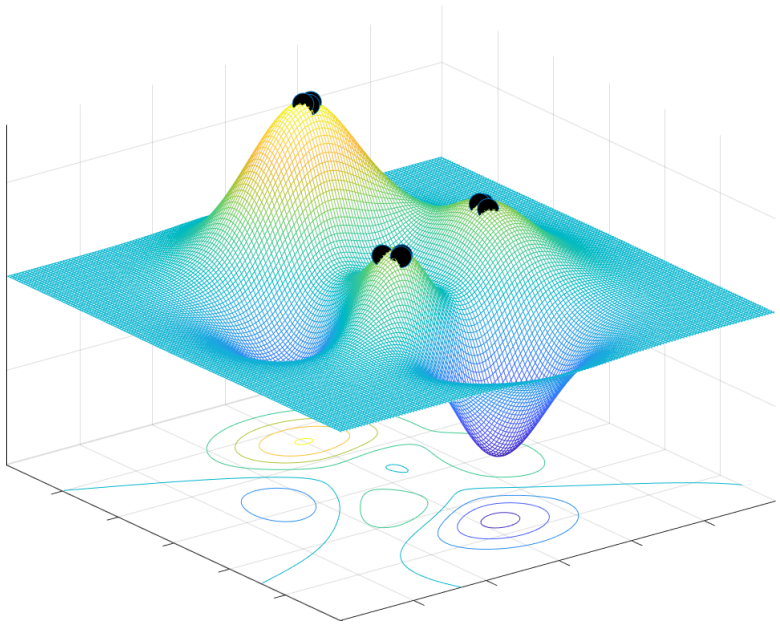

Figure 4.8: Refinement of solutions found by performing refinement optimisation on the example in Figure 4.7. A second round of optimisation on the previously found Pareto front can improve the quality of solutions.

efficient exploration of the search-space can be achieved, and as a result a faster convergence rate and a higher chance of completeness can be achieved. There is no standard formula for selecting the number of initial seeds; however, a larger population and iteration number can lead to better results.

4.5.4 Refinement of the Pareto Set

Global optimisation techniques are known to have low precision, and the optimal results found by these techniques are usually located near, but not exactly on a true optimum. For this reason a refinement of the Pareto set needs to be performed. This can be achieved with either local or global optimisation algorithms (Sections 2.2.3 and 2.2.3) and can potentially improve the quality of the previous solutions. For each of the solutions in the Pareto front (found in the previous step), a new experiment can be performed by using them as the initial population set.

4.5.5 Post Optimisation Analysis

This step is mentioned as the last one but could be executed after any of the previously mentioned steps. The objectives of this step are: (1) to extract information that can be useful for future experiments, or can be applied to similar robot design studies, (2) get a deeper understanding of the behaviours (such as patterns) and the inherent characteristics of the mechanisms that promote them, and (3) to select the designs with the highest overall merit based on a set of appropriate evaluation criteria.

After obtaining a refined Pareto set, the data can be evaluated and sorted based on criteria for desired performance. As described in previous sections, a multi-objective problem does not result in a single optimal design, but in many Pareto optimal designs. Due to a possibly large number of performance objectives, comparison or analysis of the results might not be trivial. For this reason proper evaluation criteria that reflect a design's merit must be used, and results must be analysed from multiple points of view.

Visualisation Tools

An efficient way of getting an overall idea of the results is visualisation. Proper visualisation can help make better comparisons, but can also allow the designer to identify potential patterns in the data. Nowadays, there is a variety of visualisation tools and techniques, which can represented the data from different perspectives. The following charts are specifically useful for visualising data for robot design.

- *Design Charts*.
  - *Bubble and scatter charts* help to visualise the Pareto front of designs, possible relationships between variables, coverage of the search space and effectiveness of input variable bounds.
  - *History charts* show the evolution of input/output variables over the iterations of the optimisation algorithm. These charts are useful during the optimisation experiments and can help detect convergence.
  - *Broken constraints chart* is usually a pie chart with each slice representing the percentage of each constraint that is violated over the unfeasible designs. This chart can help identify what are the limitations of the examined mechanism and its behaviours, and as a result reevaluate assumptions and decisions in the model (e.g., relax or tighten input variable bounds or update the model with better components).
  - *Parallel Coordinates chart* or *Spaghetti chart* helps to analyse high-dimensional input data. This chart allows the designer to select designs based on design or behaviour parameter specifications, and can be useful during the final design process (CAD).
- *Distribution charts* enable the designer to examine the distribution of the input/output variables. This is another way to evaluate if the algorithm has made an efficient exploration of the search space, by looking at statistical parameters such as mean, variance and more. These charts are commonly used in robustness analysis (Section 4.5.6).
- *Statistical charts*, which enable the designer to identify correlations between variables. These charts are useful for sensitivity analysis experiments (Section 4.5.2).

Multivariate Analysis

Multivariate analysis tools can reduce the data size, and help to interpret and identify patterns in the data. Useful tools for data analysis in robot design are the following.

- *Clustering algorithms*, which divide the data in clusters with similar characteristics. These algorithms are useful for identifying patterns in distinct groups of data with similar features.
- *Dimensionality reduction methods*, for removing redundant dimensions of the data for better analysis, such as Principal Component Analysis (PCA).

An example of multivariate analysis for robot design can be applying a clustering algorithm method to the behaviour data. This approach can help to identify different strategies for achieving a performance objective, which would be otherwise very difficult to detect in high dimensional spaces. Achieving a performance with various different strategies provides extra versatility in the design. Some strategies may achieve the same performance objective in better ways under different circumstances (e.g., more energy efficient or following a shortest route).

Evaluation Criteria

To facilitate the comparison between different designs, and select the best design to be built, a set of evaluation criteria must be devised based on the applications of the robot.

In cases of experiments with a large set of performance objectives where similar tasks are performed, objectives can be grouped (e.g., by adding their objective values) into behaviour *families* before being compared. This method can result in an easier comparison. Special attention must be given to the consistency of the units of the objectives that are combined. Moreover, additional evaluation criteria that were not originally performance objectives can be devised or selected. These criteria could be another measure of the robot's performance, which was not a priority during the optimisation process, but can give additional merit for the final selection.

The previous steps in the post-optimisation analysis were concerned with improving the qualitative information of the results. Having obtained useful information, the next step is to select the best design to build. The selection process can be guided by different approaches, which are dependent on the robot's applications and the available resources for building it. Examples of selection criteria are:

- design and manufacturing criteria,
- robustness criteria, and
- performance criteria.

Design and manufacturing criteria—some designs or components might be more difficult to obtain or build due to manufacturing limitations. In addition, even for the most meticulous design studies, there might be fundamental errors in the model. For example, it might be discovered during the CAD

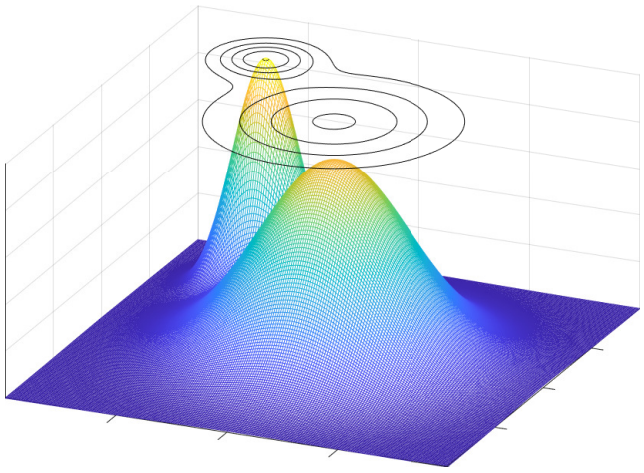

Figure 4.9: Robustness analysis identifies solutions that are robust to random variations in the input variables. In the presented figure two optima exist out of which, one is a global optimum. One can notice that the global optimum rapidly changes with variations of its input parameters, which means that uncertainties in the real world could possibly result in a significantly deteriorated performance. The local optimum has a more flat peak, which makes it more robust, and hence despite the fact that it has a lower maximum value it might be more preferable in applications where consistency and robustness are essential.

design that some parts collide with each other. This is a disastrous scenario because the selected design cannot be built (at least not without modifications that may have an effect on performance). Fortunately, the Pareto set consists of more designs, and a feasible design to build can be selected at the cost of some other measure. This choice can be aided with visualisation tools such as the *Spaghetti Chart*.

Robustness criteria—is to select the design with the best robustness and performance trade-off. This means that robustness of the design is preferred over performance (which may still be important). The selection of the final design should be based on this criterion when the application is critical or there is a high risk. An example of a critical application is surgical robotics, and a high-risk application can be dynamic robots that can potentially harm themselves or their environment.

Performance criteria—the driving factor behind the selection is solely performance. In cases where performance is essential and no major risks or limitations are faced for the manufacturing of the various mechanical parts, the design with the highest performance can be selected. For example, industrial arms need to work as fast as possible, because higher performance means higher profits.

4.5.6 Robustness Analysis

Computational models and manufactured components are not perfect. Tolerance and modelling errors are quite common, and manufacturing precision comes with a high cost. In addition, when the real robot is built and experiments are performed, assumptions such as initial conditions can never be met exactly. Moreover, classical optimisation approaches tend to over-optimise the model, which can make errors have an even higher impact. As a result, designs that have a great performance in theory, might end up performing poorly in reality.

A way to mitigate the impact of all the aforementioned is to find designs that are robust to such cases. These designs remain insensitive to acceptable and expected variations of their operating environment or input variables with minimal loss of functionality. The way to achieve this is via Robustness Analysis Optimisation (Ben [7]).

Robustness analysis can: (1) be an extra merit score on the Pareto optimal designs, or (2) be part of the main optimisation experiment (e.g., robustness is a design criterion of high importance).

1. **Merit Score**—in the case where robustness serves as an additional merit score it can be applied on the Pareto front in the following way.
   (a) Select a portion (or all if resources allow) of the Pareto front optimal designs.
   (b) Identify important parameters that are expected to have uncertainty in the real world. The selection of these parameters shall be mainly based in: (1) their importance (e.g., parameters that have a large impact in the performance of the robot), which is already quantified from the sensitivity analysis step (see Section 4.5.2), (2) manufacturability precision of parts, and (3) available resources.
   (c) Define stochastic input variables and sampling algorithm. For each examined design generate random samples (called robust designs) in its close vicinity. From these designs statistical data are gathered to evaluate the robustness of the examined design.
   (d) Define new objectives. Minimise the variance (or another statistical measurement such as mean, median, maximum or minimum value) of each objective, and use this value as an extra evaluation criterion for each design.
   (e) Define constraints. The constraints set a limit on how much an objective value is allowed o vary.
   (f) Run the optimisation.
2. **Performance Objective**—if the designer wishes to incorporate robustness as an optimisation criterion in the optimisation process, then the optimisation experiment can be modified in the following way.
   (a) Define stochastic input variables and sampling algorithm.
   (b) Define objectives and constraints.

(c) For each design generated by the global optimiser, a set of robust designs is sampled from a predefined distribution, which are also evaluated.

(d) Run the optimisation.

The selection of the distribution that the robust designs are generated from is up to the designer. For example, a designer can select to model uncertainty with a normal distribution (Gaussian) that has a given standard deviation and a mean equal to the parameter values of the original design. In this example, the uncertainty is determined by the standard deviation. Similarly to the previous steps, an application of this step is presented in the case study in Section 6.6.7.

4.6 Conclusion and Discussion

This section concludes the first part of this thesis. This part presented the main contribution of this work, which is a framework for the design and behaviour co-optimisation of high-performance mobile robots, together with a proposed optimisation methodology. The aim of this work is to facilitate the design process of high-performance mobile robots by investigating the intrinsic relationship between a design and its behaviours, by using realistic models of the robot's mechanism and limitations, and by thoroughly defining its performance objectives.

The first chapter of this part presented an introduction to the field of mathematical optimisation and its most important concepts and definitions. This included a discussion on the various types of optimisation problems, their challenges and applications, as well as the various types of algorithms that exist for addressing these problems. In the following chapter, related work in the field of robot design optimisation was presented together with a critical evaluation of the field.

The final chapter is dedicated to the main contribution of this thesis. It consists of two parts, (1) the optimisation framework, and (2) the optimisation methodology. In the latter part an abstract model of the optimisation architecture is presented and a discussion on the basic concepts of the proposed design philosophy is presented. In the former part an optimisation methodology is proposed for performing the optimisation experiments.

In more detail, a two-layer optimisation framework for the design and behaviour co-optimisation was presented. The overall process is a two-layer multi-objective optimisation scheme in which the upper layer (the mechanism layer) searches the space of mechanical design parameters, and the lower layer (the performance tester layer) tests each design against each one of a list of performance objectives, providing a separate score for each objective. Almost every test is itself an optimisation problem in which the optimiser searches a space of behaviour parameters looking for the best behaviour of the given mechanical design to accomplish the given performance objective. This is an example of mechanism and behaviour co-design.

The novel aspects of this work are that a large repertoire of performance objectives is considered, many of which conflict with each other, and that a high level of performance is sought, that is close to the limits of what is physically possible with today's technology. The latter, in particular, requires

the use of accurate dynamics simulations based on a detailed dynamic model that accounts for all relevant physical limitations of the robot, such as speed and force limits, energy losses due to friction and hysteresis, and so on.

For the purpose of performing the optimisation experiments in a way that can improve the quality of the results, an optimisation methodology is proposed. Specifically, the methodology consists of six parts which are: (1) sensitivity analysis to identify the most important design parameters, (2) preliminary design exploration with DOE techniques to obtain qualitative information from a small sample of the design space, (3) rough optimisation to obtain rough optimal solutions, (4) refinement of the Pareto front to improve the precision of the solutions, (5) robustness analysis to obtain designs that are invariant to the real world imperfections, and (6) post-optimisation analysis to evaluate and select the best design. Note that the order of the aforementioned steps does not need to be as listed, and the process can be repeated until the desired results are obtained.

4.6.1 Challenges

The challenges faced with the proposed design philosophy involve both theoretical and practical aspects, and are summarised below.

1. Multiple and conflicting high-performance objectives—in the analysis of extreme behaviours the performance objectives are close to the physical limits of the robot's components. Moreover, the performance objectives can be of conflicting nature, which results in a Pareto front optimisation (Section 2.2). Finally, the decision maker must select the design with the most acceptable performance compromise (between the evaluation criteria).

2. Realistic modelling of the robot's mechanism and its limitations—requires deep understanding of real robotic systems, which includes knowledge of dynamics, control theory, mechanical and electrical engineering, sensing technologies, modelling and simulation, software and more.

3. Low cost of manufacturing and testing—for building realistic models obtaining experimental data is essential. However, obtaining these data might require test-rigs and also specialised equipment such as Universal Testing Machines.

4.6.2 Advantages

The proposed optimisation framework and approach could have the following advantages.

1. Faster and less expensive robot production—with a more complete optimisation study, fewer experimentation and design cycles are required. Modelling software and simulations require time to be developed; however, this can be compensated by the fact that fewer physical experiments will be required.

2. Prototype performance closer to design expectations—realistic software can reduce the gap between simulation and reality.

3. Design of versatile robots—by optimising for a large repertoire of behaviours, robots can have a more generalised purpose and might be able to assist humans in various scenarios.

4. Unprecedented performance—experiments in simulation are significantly cheaper in terms of time and money and have fewer risks. This means that much more complicated experiments can be executed in simulation, which results in larger search-space exploration that increases the chances of discovering extreme behaviours. However, a realistic simulation that takes into account the robot's limitations is essential in achieving the aforementioned.

5. Robust behaviours—realistic modelling of limitations results in co-evolution of behaviour and mechanism so the robot can operate to its maximum potential, which is close to its limits, but not beyond them. Being aware of its own limits, the robot will not try to surpass them, and hence have less risk of damaging itself.

4.6.3 Limitations

The presented approach may have the following limitations.

1. Development of new software—detailed models require new software to be developed, tested and documented.

2. Sophisticated optimisation software—multi-objective global optimisation problems are difficult to solve. This means that sophisticated algorithms are needed to solve them. Open source software for optimisation exists, but commercial software tends to be more reliable and comes with better technical support. In addition, efficient tools for post-optimisation analysis are also required.

3. Large computational cost—as model complexity increases so does the software, which might require substantial resources in terms of time and memory.

4. Real experiments—to model the various parts of the robot dedicated equipment (e.g., Universal Testing Machines) might be required, or new test-rigs to be built (which requires a workshop equipped with various tools and machinery).

4.6.4 Generalisation

This framework and the proposed optimisation methodology were designed for facilitating the design process of high-performance mobile robots. This approach is based on a systematic and complete study of a robot's mechanism, its behaviours and its limitations, which uses mathematical models and concepts to achieve a maximum physical performance. Even though it was only tested on the design of a monopod hopping robot, the approach can be applied to other types of robot. The optimisation approach was designed to be agnostic, and it is not concerned with what happens inside the simulation.

Specifically, the proposed optimisation framework and methodology can be applied to any design optimisation problem if the following are provided:

1. a simulated model (could be a very detailed realistic model or even a simple kinematic model) of the design and its behaviours, and

2. a properly defined set of performance requirements and constraints.

The proposed optimisation process is based on global optimisation techniques that make very few assumptions about the system to be optimised. This means that the overall process can also be applied in black box models.

In the next and second part of this thesis the proposed optimisation framework and methodology are applied for the design optimisation of a one-legged hopping robot. Detailed models of the mechanism are presented together with a detailed optimisation process. In addition, results are presented and compared using various post-optimisation techniques.

Part II

Case study: Skippy, a Hopping Monopedal Robot

Chapter 5

Building a Model: Hardware and Behaviour

5.1 Introduction

The aim of this part is to demonstrate the effectiveness and power of the proposed optimisation approach by applying it to the design optimisation of a high-performance monopod robot called Skippy. This part is divided into two separate experiments with different aims and objectives. Specifically, the second part is a more advanced and an in-depth continuation study of the first part. In both experiments the same robot (Skippy) is used, with the same models for its mechanism (except the model of the springs), behaviour and simulations; however, the optimisation experiments and their objectives differ.

Skippy is a highly athletic monopedal robot that is able to make high hops and balance skilfully in 3 D. It is designed to be light and powerful enough to perform impressive athletic feats such as as vertical hops up to 3 m, which is close to its performance envelope. As mentioned in the previous chapter, properly defining what the robot is built to achieve can aid the design process so the final prototype displays a performance close to the mechanism's true potential.

Skippy is also designed to be robust enough to survive crash-landings from failed hopping attempts, and it is simple enough (only 2 actuators) to allow a thorough analysis of its mechanism and to facilitate real experiments. Simplicity and the high-performance requirements make Skippy an ideal robot for proving the effectiveness of the proposed optimisation approach.

Skippy is part of the Skippy project [33] which aim is to prove the effectiveness of a systematic and thorough design study by demonstrating unprecedented behaviours. This can be achieved by utilising the maximum potential of today's technology and driving it to its limits. Unlike other robots such as biologically-inspired, Skippy is technology-inspired—this means that Skippy's design does not mimic any other biological creature but is driven by the potential of current available technology. Moreover, Skippy's simplicity allows a deeper study of legged locomotion as well as balancing because its one leg forces the robot to continuously strive to balance, while performing any other task.

The objective of the optimisation study is to discover a single design capable of meeting several high performance objectives. The robot is required to perform a large repertoire of behaviours, which are: increasing height vertical hops, decreasing height vertical hops, travelling hops, a somersault, have a high balancing ability and be energy efficient. The performance objectives were defined in a series or preliminary experiments, and except the robot's balancing ability, the rest were set to specific values (e.g., reach a hopping height of 3 m). The result is a single design with a vector of evaluation scores that reflect the robot's ability in each of the desired requirements.

This chapter starts with a literature review on the various existing hopping robots, and a critical evaluation on their designs, their performance and their design objectives (i.e., what was the purpose of the design, and what did it eventually achieve).

Thereafter, the proposed approach of Part I is applied for the design and behaviour co-optimisation of Skippy. The first step is building a model of the system. This part consists of the following sub-parts: (1) model the mechanism (Section 4.2.2), (2) model the behaviour (Section 4.2.3), (3) define the performance envelope and the objectives (Section 4.2.4), and define the constraints (behavioural and mechanical) (Section 4.2.3). Following that is the determining the problem type step (Section 4.3), and software selection step (Section 4.4), and examples of the simulated models are presented.

Next, a set of preliminary experiments are presented in an existing robot, called Tippy, which is the predecessor of Skippy. This study demonstrates the balancing algorithm and the balancing performance measure that will be used on Skippy. The purpose of this section is to justify the effectiveness of this measure, which is called the Angular Velocity Gain (AVG) (Featherstone [30]).

Over the next sections the optimisation methodology is applied to the design of Skippy. The study is divided into two parts. In the first part a design that is capable of performing increasing height vertical hops, decreasing height vertical hops, travelling hops, and demonstrating a high balancing skill is sought. Due to the multiple demanding performance objectives and the fact that they are of conflicting nature, makes this problem difficult to solve. Prior to the study presented in the first part there was no Skippy design capable of achieving all of these objectives. By applying the proposed approach a set of Pareto optimal Skippy designs capable of achieving all of the aforementioned objectives was discovered. The results of this study were used as seeds for a second and more advanced study that included also a somersault, a more realistic model of the springs and more advanced evaluation criteria for the performance of the designs. For both experiments the proposed optimisation framework and methodology was applied. Finally, performance comparisons, example behaviours and various results from the experiments are discussed and presented.

5.2 Related Works: Hopping Robots

Legged locomotion, and specifically hopping robots, have been the topic of interest since the early 1980s when Raibert introduced his 2 D and 3 D balancing and hopping robots [83]. Legged robots are of particular interest due to the great mobility offered by their legs, which allows them to traverse cluttered terrains, something that wheeled robots cannot do. A big part of the world of humans

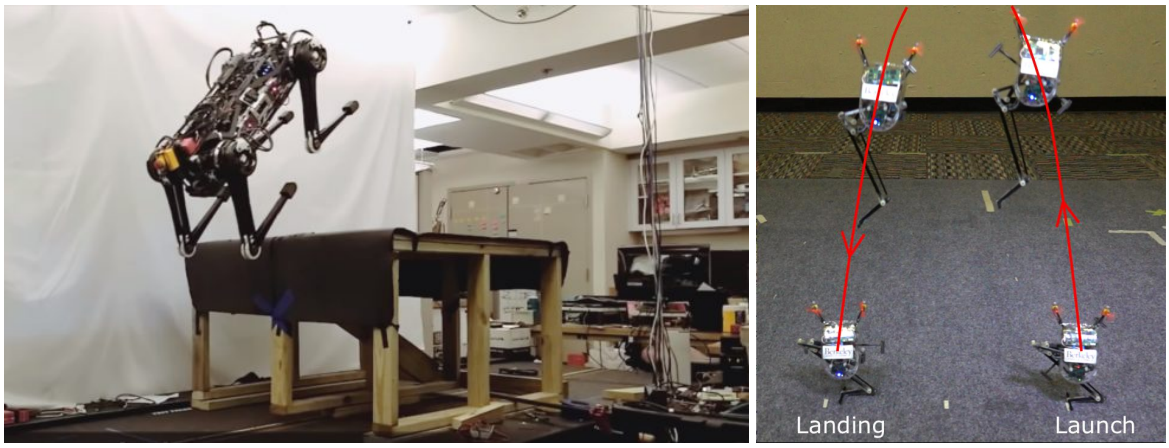

Figure 5.1: Left: MIT's cheetah 3 jumping on a 0.76 m desk [9]. Right: Salto-1P performing a targeted jump [111].

consists of environments full of obstacles, tight spaces, stairs and more, and hence legged robots can find many uses not only in everyday scenarios, but also exploration, disaster scenarios and more.

Since Raibert's hopper, several years passed until noticeable progress was observed in legged locomotion, and specifically in hopping and jumping. Scarfogliero et al. [95] designed a frog-inspired robot, named Grillo capable of achieving travelling speeds of 1.5 m/s. Their robot had a length of 5 cm and a total mass of 15 grams and used feed-forward signals to achieve its objectives. Li et al. [62] achieve 0.1 m height and 0.2 m length leaps with an insect-like robot of 0.22 kg. The minitaur robot [57] achieves a 0.48 m vertical jump using direct drives. Direct drives are more robust than geared drives, because the latter are susceptible to damage during high-force impacts. Hopping is a highly dynamic motion that results in large forces, and for this reason also Skippy does not use gears.

Ha et al. [45] co-optimised the design and motion of a single-legged hopping robot for the single objective of reaching a 1 metre vertical hop from a resting configuration. In their work, link lengths are optimised using an Evolutionary Algorithm. Saar et al. [92] presented a hopping robot that achieved a travelling speed of 0.26 m/s. Shamsah et al. [99] achieve hopping on a monopod, a tailed monopod and a tailed biped. The former achieved a jumping height of about 0.3 m and has a mass fo 2.77 kg. It uses a 4-bar linkage and a carbon-fibre leaf spring similarly to Skippy, which also uses a 4-bar linkage, but a glass-fibre leaf spring instead.

Siedel et al. [97] also studied one-legged machines, where a regression model for hopping is built using experimental data. Their miniature hopping robot has a length of 0.25 m and hops onto obstacles of 2 cm height. In this case, even though their robot was designed for hopping, it appears in their work that its hopping performance was only evaluated in the real robot and was not part of the design process. Batts, Kim and Yamane [5] present a monopedal hopping robot capable of continuous 3 D hopping. Their robot has one motor, two springs in parallel and uses Raibert's [83] hopping and balancing controller; however, it must continuously hop to maintain its balance.

Verstraten et al. [107] also achieve low hopping cycles of 2.4 cm with a 2.7 kg tethered robot. Bledt et al. [9] use QP to find feed-forward behaviours for achieving a 0.76 m hop on and off a surface with the *Cheetah 3* robot. Grimminger et al. [41] use QP to discover a feed-forward signal for achieving a 0.65 m jump on a 2.2 kg 3 D printed quadruped. Similarly Katz et al. [56] also use QP for behaviour optimisation to achieve a 0.3 m back-flip with a feed-forward signal on the mini-cheetah. Their robot weights 9 kg and the trajectory for the back-flip is generated offline. Also, their robot can achieve gaits of speeds up to 2.45 m/s. These works do not state if any of the displayed behaviours were taken into account during the design process of their robots.

In the work of Yim et al. [111], high-performance hopping and balancing ability is demonstrated with the same angular momentum-based balancing algorithm (Featherstone [31]) that Skippy will use. The performance of their miniature hopping robot (Salto-1P) is measured via experimentation. With a weight of 0.11 kg their robot can achieve a hopping height of 1.25 m. Thanks to precise angular momentum control, the robot displays a high jumping precision, with an error standard deviation of 1.6 cm in targeted jumps. .

This section presented several robots of various sizes and masses that demonstrate impressive hopping/jumping skills; however, most of the presented behaviours were either the result of experimental tuning or behaviour-only optimisation. These approaches ignore the relationship between the robot's performance objectives and its design, and the final performance is the result of design choices or optimisation experiments that were not aimed to achieve the mechanism's true potential in the examined behaviours. This means that the presented mechanisms have not been properly explored for optimality in the tasks that they are required to do.

Furthermore, the presented studies optimise for a small set or even only one desired behaviour. However, Skippy is optimised for a large repertoire of behaviours with the aim of allowing it to travel, hop at different heights, perform complicated acrobatic motions such as a 2-metre triple somersault, as well as to be a skilled balancing machine. To have practical use in the lives of humans, robots must display a plethora of behaviours, which must be taken into account during their design process.

Finally, the majority of the presented legged robots are relatively light robots (less than 10 kg). Even though the practical use of small robots is limited, due to their small size and mass these robots are more robust in crash landings, easier to control, experiment and build, and more agile, which means that they are cheaper to maintain and build. Experimenting in relatively smalls robots makes highly dynamic motions easier to achieve, and with less risk. This can help build the foundations for developing and achieving the same motions for heavier and bigger robots with higher potential and usability. Skippy is also designed to be as lightweight as possible, with the first prototype having an estimated mass of 3 kg.

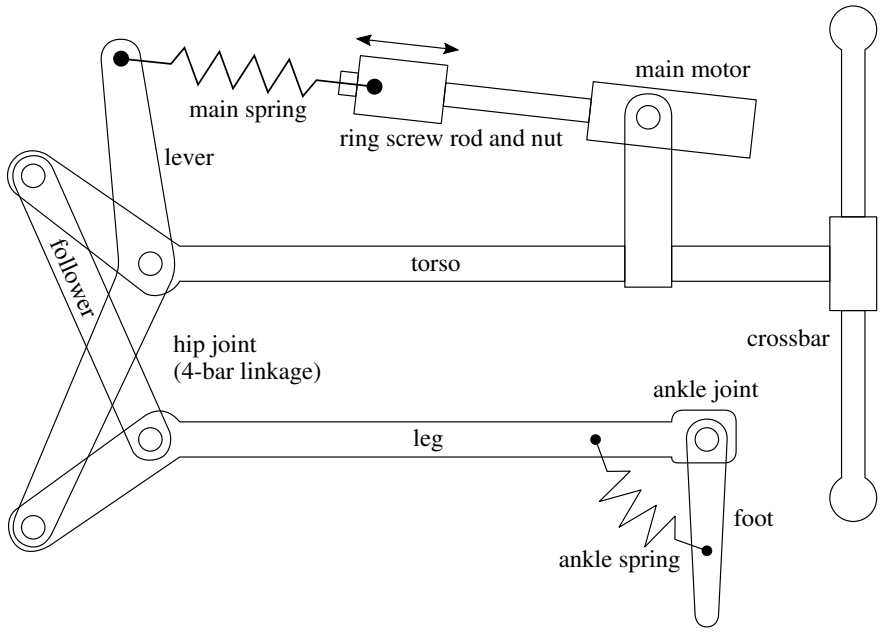

Figure 5.2: A schematic diagram showing the relevant parts of Skippy. The batteries, electronics and crossbar drive components have been omitted. The diagram is not to scale, and does not show the parts in their true shape.

5.3 Building a Model

In this section the model of Skippy is presented. Following the approach of Section 4.2, the mechanical model of the robot is first defined. Its kinematic and dynamic model are defined, and its various components and limitations are listed. In addition, various design choices are justified and some details about the realistic models are presented. The modelling of the behaviour is explained together with the behavioural constraints, assumptions and challenges of the modelling process.

The scientific objective of the Skippy project is to test the hypothesis that the gap between the relatively poor physical performance of legged robots and the much better performance suggested by the specifications of the available component parts is mainly due to poor design. To this end, the Skippy project sets out to design, build and demonstrate a high-performance robot that is so mechanically simple that it is feasible to design it more thoroughly in order to achieve higher physical performance. This is why Skippy has only one leg (simplicity), only two actuators (simplicity), and aims to hop to a height of 3 m, make high somersaults, and exhibit other similar behaviours (high performance).

5.3.1 Hardware

Figure 5.2 shows a schematic diagram of Skippy's mechanism, springs and main motor. Certain parts have been omitted, including the crossbar motor and transmission. The crossbar itself rotates about the long axis of the torso, and is used to balance and steer the robot in 3 D. (See Azad [3] and Azad

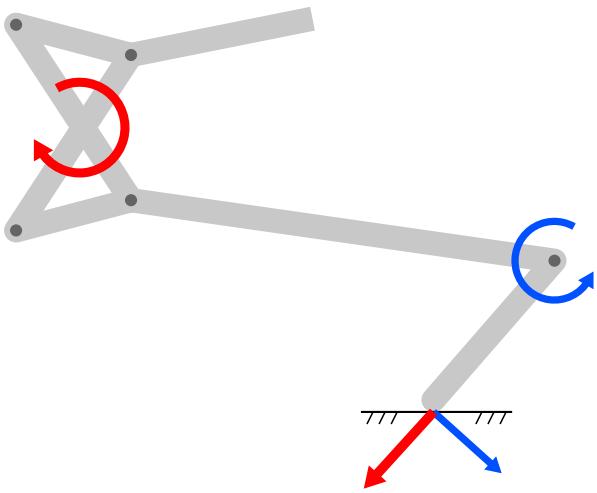

Figure 5.3: Torques at the hip and ankle joints push Skippy's foot into the ground in two different directions.

and Featherstone [4] for a description of how this is done, and [33] for animations of 3 D balancing and steering.) For the 2 D study in this thesis the crossbar is treated as being part of the torso. Apart from the crossbar, the rest of the mechanism is planar. A more detailed diagram of Skippy, showing most of its kinematic parameters, appears in Figure 5.4.

The main motor is responsible for hopping and for balancing in the robot's sagittal plane. Its output shaft is connected directly (i.e., without any intervening gear) to a ring screw rod. The ring screw [32] is a recently invented mechanical component that performs the same function as a ball screw but is able to operate at much higher speeds. The main motor operates the lever via the ring screw and main spring; and the lever in turn operates the leg via the 4-bar linkage, which is the hip joint. Finally, the leg is connected to the foot via a spring-loaded revolute joint at the ankle (see Section 5.4.1). The need for an ankle joint was established early in the Skippy project as the solution to the following problem: Skippy must have a specific linear and angular momentum at lift-off if it is to perform a hop of a given height and travel, and land on its foot at the end, and this must be accomplished using only one actuator. For this to be possible, the actuator must be able to influence both the magnitude and the direction of the ground force (the force transmitted from the foot to the ground, which is the opposite of the ground reaction force). As shown in Figure 5.3, torques at the hip and ankle joints produce ground forces in different directions. If the ankle joint is removed then the hip actuator can influence only the magnitude of the ground force, which is not enough. If both joints are present and actuated then Skippy has full control over the magnitude and direction of the ground force, and can therefore make any kind of hop. If, instead, both joints are present but only the hip joint is actuated while the ankle joint is spring-loaded, then varying the hip joint torque affects both the magnitude and direction of the ground force, and some of our earliest simulation experiments (unpublished) showed that this was enough to allow Skippy to perform a large repertoire of hops.

Bodies of the robot	Joint variables of the robot
B_1 to B_3—fictitious massless bodies that give the foot the freedom to lift off the ground and roll	$q_1 + q_3$—define the x coordinate of the toe-ground contact point
B_4—foot	q_2—y coordinate of the toe-ground contact point
B_5—leg	q_4—foot angle (combines with q_3 to make a rolling contact)
B_6—lever	q_5—ankle angle
B_7—torso	q_6—leg-lever angle
B_8—follower	lever-torso angle
B_9—rocker (carries the main motor and includes the screw rod)	q_8—torso-follower angle
B_{10}—coupler (includes the ring screw nut and main spring)	q_9—torso-rocker angle
	q_{10}—nut displacement (towards motor)

Table 5.1: Left column: the list of bodies in Skippy's model; right column: the list of joints in Skippy's model.

Kinematic and Dynamic Model

A more detailed kinematic model of Skippy is presented in Figure 5.4. As explained earlier, the real robot will also have a second motor to control the crossbar, which is used to balance and steer the robot in 3 D (see Section 5.5). Because this motor does not need to be actuated during the behaviours examined in this thesis, the model includes only their masses which are added to the torso. Table 5.2 contains some of Skippy's kinematic and dynamic parameters, as well as kinematic limits. In addition to these, there are more parameters that are presented in the rest of this section. These parameters are coupled with other components and are defined with them (e.g., the thrust bearing force limit of the ring screw bearings, or the kinematic parameters of the 4-bar linkage).

Table 5.1 lists the bodies and joints of Skippy's model, together with their description. The independent joint variables are q_1, q_2, q_4, q_5 and q_{10}. q_4 controls q_3 (rolling contact with ground), and q_{10} controls q_6, q_7, q_8 and q_9 (4-bar linkage and rocker). Joints 1 and 2 come into play when the foot leaves the ground or when the foot slips while rolling. As the simulations assume rolling without slipping, it follows that $q_1 = q_2 = 0$ at all times. The robot is folded both in the open-loop zero position (all variables zero—see Figure 5.4) and the closed-loop zero position (all independent variables zero), with the foot vertical and the leg and torso shafts horizontal.

4-bar Linkage

The 4-bar linkage serves two purposes in Skippy. First, it amplifies the motion of the lever, so that the lever needs to turn through only 90° in order to make the leg turn through 180°, which is its full range of motion. Second, it creates a nonlinear relationship between the lever angle and the leg angle such that the leg can exert large forces at low speeds when the robot is folded (as shown in Figure 5.2), and less large forces at progressively higher speeds as the robot unfolds. The analysis does not consider

Name	Description	Value	LB	UB
Kinematic Parameters and Limits				
rtoe	radius of toe	0.03		
d2toe	length of foot (body 4)	∼	0.22	0.26
p2x	x coordinate of P2 (ankle) relative to P1	0.55		
p2y	y coordinate of P2 (ankle) relative to P1	0.05		
A3	angle of 4-bar segment AB within torso	0.35		
d05	lever length	∼	0.1	0.11
A5	lever angle	0.45		
p6x	x coordinate of rocker joint in torso	0.4		
p6y	y coordinate of rocker joint in torso	0.07		
Dynamic Parameters				
foot_m	mass of foot (body 4)	0.25		
foot_cx	foot centre of mass x coordinate	0		
foot_cy	foot centre of mass y coordinate	0.11		
foot_rog	foot radius of gyration	0.09		
leg_m	mass of leg (body 5)	0.35		
leg_cx	leg centre of mass x coordinate	-0.25		
leg_cy	leg centre of mass y coordinate	-0.02		
leg_rog	leg radius of gyration	0.22		
lever_m	mass of lever (body 6)	0.25		
lever_cx	lever centre of mass x coordinate	0		
lever_cy	lever centre of mass y coordinate	0.1		
lever_rog	lever radius of gyration	0.08		
torso_m	mass of torso+, rocker, nut (bodies 7, 9, 10)	2		
torso_cx	torso+ centre of mass x coordinate	∼	0.36	0.44
torso_cy	torso+ centre of mass y coordinate	0.025		
torso_rog	torso radius of gyration	0.18		

Table 5.2: List of 25 dynamic and kinematic parameters of Skippy's mechanism (the list does not contain the 4-bar linkage parameters). Parameters with tilde in their Value field are optimised, and hence do not have a fixed value. LB and UB stand for lower and upper bounds respectively. Fields LB and UB are empty for constants. Lengths are in metres, masses in kilograms and angles in radians. The symbol 'torso+' means the torso plus the crossbar, treated as a single rigid body.

Name	Description	Value
bar4_a	length of 4-bar segment (in torso)	0.04835
bar4_b	length of 4-bar segment (in lever, body 6)	0.122
bar4_c	length of 4-bar segment (in leg)	0.04988
bar4_d	length of 4-bar segment (follower, body 8)	0.1179
bar4_phi	CD offset angle at motion limits	0.1301
bar4_Bmin	4-bar angle at maximum stretch	0.3699
bar4_Bmax	4-bar angle at maximum fold	1.9407

Table 5.3: List of kinematic parameters of the 4-bar linkage. Lengths are metre and angles in radians.

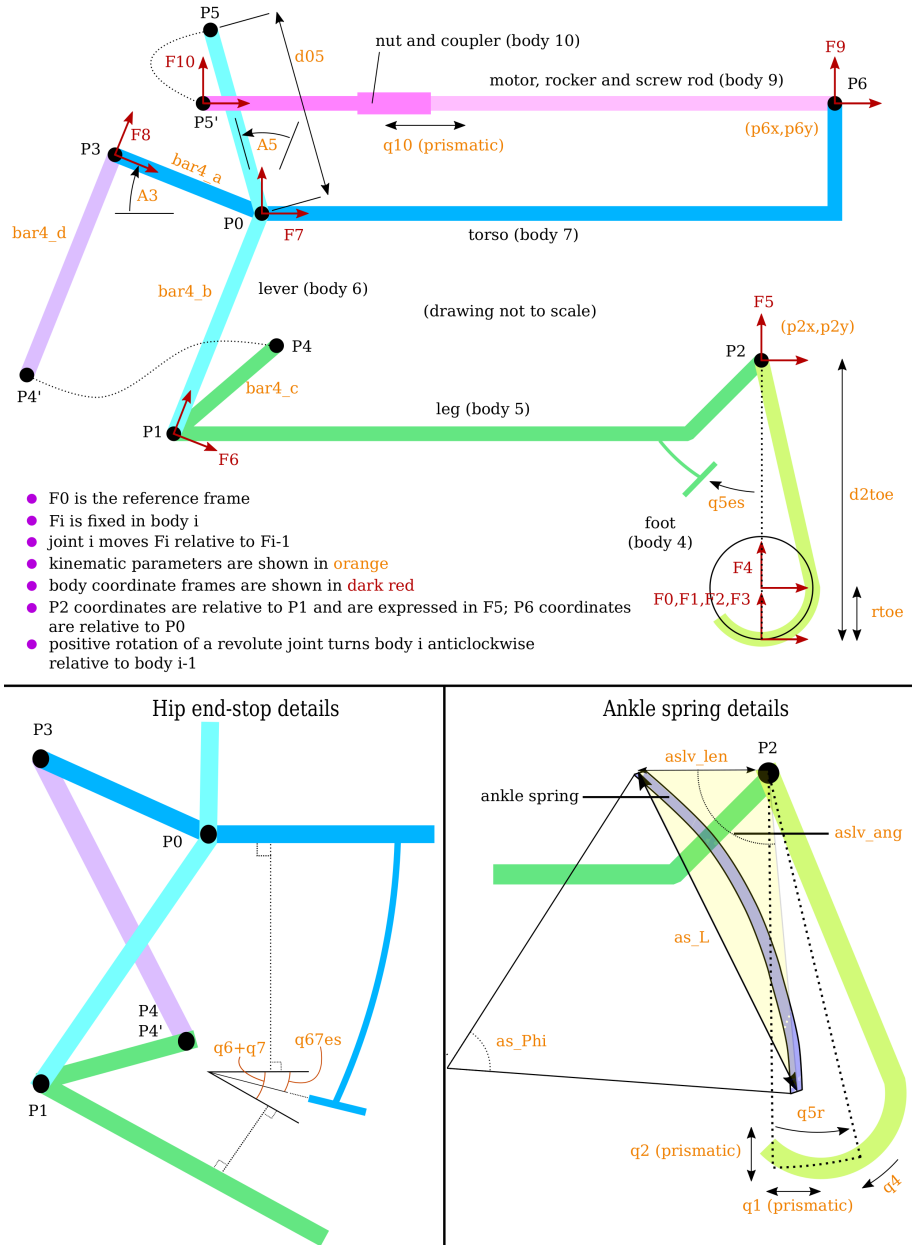

Figure 5.4: Top diagram showing Skippy in its open-loop zero position (every joint variable equal to zero), plus details of the hip end stop and ankle spring. In Skippy's closed-loop zero position (not shown), P4' coincides with P4, P5' with P5, the lever is rotated slightly clockwise, but the leg is still parallel to the torso (q_6+q_7=0). Note that Skippy is pressing into its hip end stop in its zero position. Parts are not drawn in their true shapes.

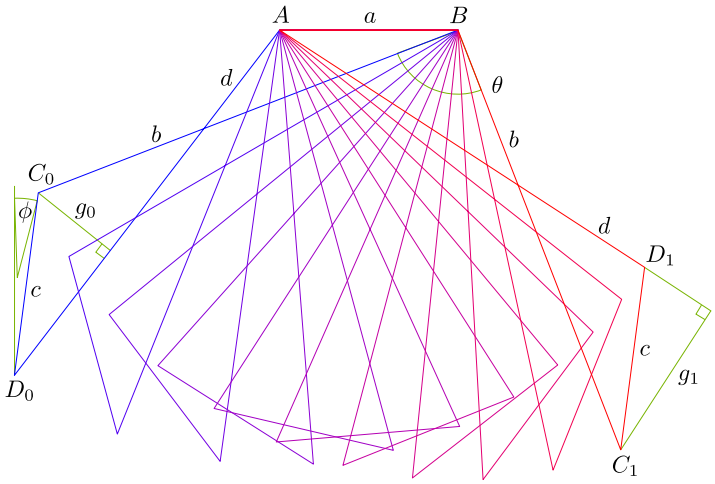

Figure 5.5: Kinematic model of the 4-bar linkage, showing a range of configurations. Parameters a, b, c, d, and ϕ are corresponding parameters bar4_a, bar4_b, bar4_c, bar4_d and bar4_phi in Table 5.3. bar4_Bmin and bar4_Bmin are the minimum and maximum value of input angle θ, which span is 90°.

the lever that connects the 4-bar with the main motor, which is optimised in the final optimisation experiment presented in Section 6.6.

Table 5.3 shows the kinematic parameters of the 4-bar linkage which is accommodated by Figure 5.5 to describe the mechanism. The kinematic parameters of the 4-bar linkage were obtained via a separate optimisation process, where the 4-bar was optimised in isolation, and is not described in this thesis, nor has not been published elsewhere. This optimisation study was performed under the assumption that the 4-bar design would serve as good initial guess, and could be the subject of subsequent optimisation studies, that unfortunately time did not allow to perform.

Figure 5.5 shows the kinematic model of the 4-bar linkage in various configurations from a fully folded to a fully extended configuration. The segment AB is fixed in the torso; CD is fixed in the leg; BC is the bottom half of the lever; and AD is the counter-lever. The lever swings through an angle θ as the leg swings through a full 180° from C_0D_0 to C_1D_1. g_0 is the perpendicular distance from C to AD, and g_1 is the lever arm of the counter-lever about the point C.

Motor

The main motor was chosen to be a Maxon DCX32L [70] for the following reasons. First, we chose Maxon because it provides detailed data on its motors, including the parameters needed by a thermal model of the motor's windings and housing. Skippy is so energetic that it could easily overheat this motor, so a thermal model (supplemented with temperature sensors) is necessary for safe operation of the robot. We then chose the most suitable brushed DC motor from Maxon's range; that is, the lightest motor that could deliver sufficient power and torque.

There are two reasons for preferring a brushed DC motor over a brushless one. First, brushless motors get their advantage over brushed motors by operating at higher speeds. Although the ring

Characteristics		
K_τ	Torque Constant [mNm/A]	27.3
S_τ	Stall torque [mNm]	1980
R_{25}	Winding Resistance at 25°C [Ω]	0.331
L	Terminal inductance [mH]	0.052
I_{nl}	No load current [mA]	164
ω_{max}	Max Speed [rpm]	11,300
τ	Mechanical time constant [ms]	3.24
I_r	Rotor inertia [gcm^2]	72.8
m	Weight [kg]	0.32
V_{nom}	Nominal Voltage [V]	24
T_{nom}	Nominal torque [Nm]	0.108
I_{stall}	Stall current [A]	72.5

Thermal Data		
R_{tha}	Thermal resistance housing-ambient [K/W]	7.28
R_{twh}	Thermal resistance winding-housing [K/W]	2.3
T_{cw}	Thermal time constant of coil [s]	42.2
T_{cm}	Thermal time constant of housing [s]	837

Table 5.4: Table with Maxon's DCX32L 70 W 24 V [70] motor data.

screw can cope with the extra speed, the consequence of a main motor running at higher speed is that we must either reduce the pitch of the ring screw or increase the travel of the nut, or do both. Unfortunately, we are constrained by a practical lower limit of 4 mm on the ring screw pitch, which means that we would have to increase the travel of the nut; but this would require several parts of Skippy to be larger, and therefore heavier, which negates the advantage of a brushless motor. A further problem with the higher speed is that it exceeds the resonant frequency of the screw rod. The second reason is that we found a good motor driver for brushed DC motors: the Pololu G2 High-Power Motor Driver 24v21 [81]. Good drives exist also for brushless DC motors [26], but are aimed at much more powerful motors.

Table 5.4 shows the parameters of the Maxon DCX32L motor that Skippy uses. These parameters are used to build the electrical, the mechanical and the thermal model of the motor. Taking into account the dynamics of the motor in the model, reduces the gap between simulation and reality, and only behaviours that can be tracked by the motor (e.g., do not surpass the motor's limits) are explored. Because a realistic model of the motor is used, the input to the plant is a voltage.

Ring Screw

A ring screw is a mechanism that achieves an almost frictionless motion between a nut and a screw rod. It performs the same function as a ball screw but at much higher speeds [32]. For this reason it can be directly coupled to the motor shaft, and avoid the use of gears which are susceptible to damage during large impacts. The ring screw is also largely responsible for transmitting the explosive power required for hopping, due its fast speed. The parameters of the ring screw are presented in Table 5.5, and details on its model, which is not presented in this thesis can be found in [32].

Name	Description	Value
pitch	ring screw rod pitch [mm/rev]	4.5
effi	ring screw efficiency	0.9
thrust	ring screw maximum (pulling) thrust [N]	1,500
stroke	ring screw stroke (max nut movement) [m]	0.15
Irsr	ring screw rod and nut inertia [gcm^2]	13.6
wmax	ring screw speed limit [rpm]	9,000

Table 5.5: Ring screw parameters.

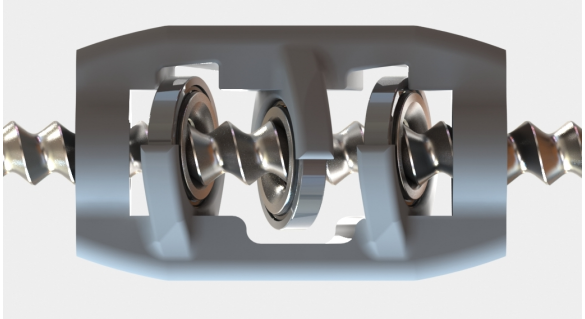

Figure 5.6: Image of the ring screw rod, nut and ball bearings. The rings are placed in such way, that a theoretically perfect rolling contact between the rod and each ring is achieved, resulting in an almost frictionless sliding motion.

A prototype has been tested at 16,500 rpm [49], which is approximately four times the speed limit for a comparable ball screw. However, the speed limit of 9,000 rpm was set in the design study in order to avoid a resonance in the screw rod. The motor's mechanical speed limit is only a little beyond that at 11,300 rpm. Skippy needs this much speed in order to meet its performance objectives. Essentially, a ring screw consists of a screw rod and a nut, and the nut contains rotating rings that make rolling contact with the rod as it moves, resulting in a nearly frictionless movement between the rod and nut. In Skippy, the main motor spins the rod, and the nut travels back and forth. A picture of the ring screw appears in Figure 5.6.

Springs

The two springs are crucial to the performance of Skippy, and so they have been the subject of much attention. In the real robot, both springs will be fibreglass leaf springs of constant thickness and varying width, which are curved in a circular arc in their rest state. Their nominal behaviour can be characterised accurately with three parameters: rest length, arc angle at rest, and force to compress by 25%. (In reality, both springs operate mainly in compression, despite the appearance in Figure 5.2 that the main spring operates mainly in tension. The figure does not show the true mechanical arrangement.) A fourth parameter, the force hysteresis, models the energy loss in real springs.

These models were built after a series of stress-strain tests we performed on a batch of real springs (see Figure 5.8). The black line in the right-hand graph in Figure 5.7 is the profile of a real

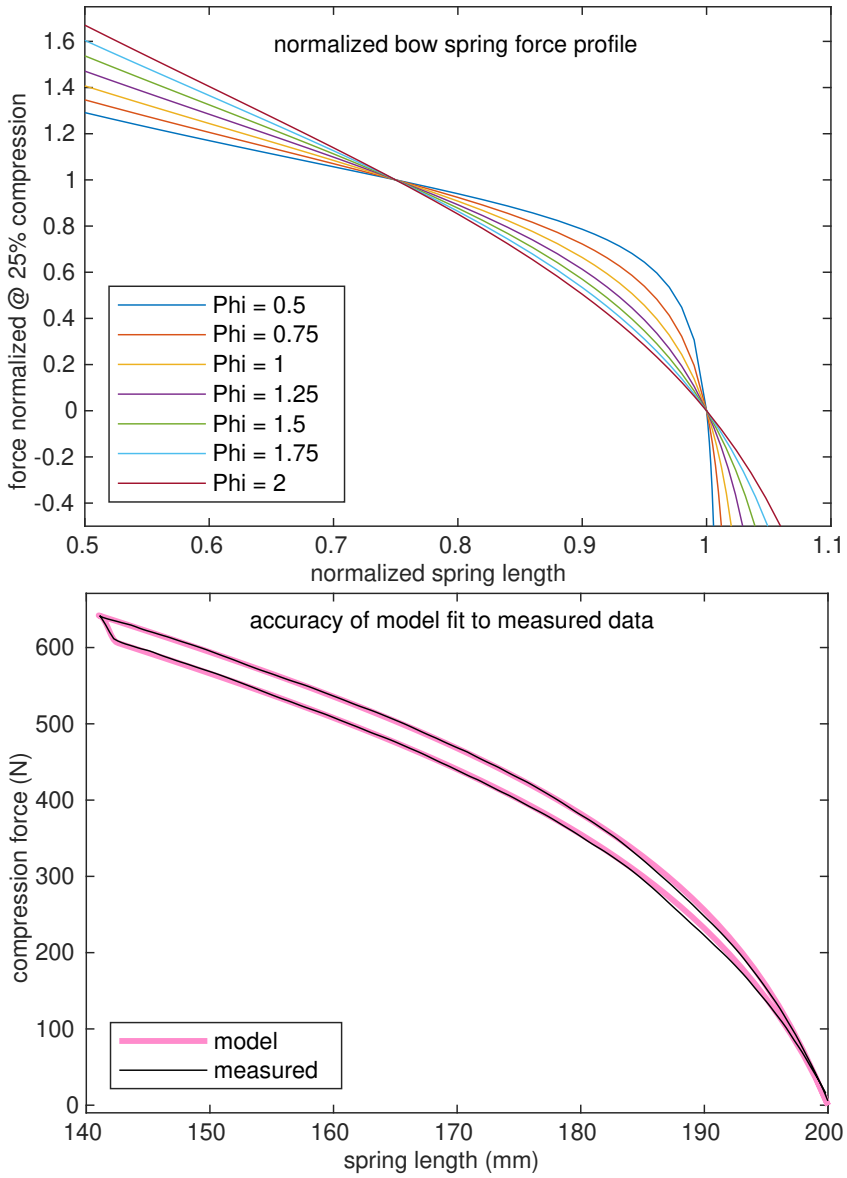

Figure 5.7: Top: A family of force profiles for tapered leaf springs with different arc angles, normalised for unit length and unit compression force at 25% compression. Bottom: Accuracy of model fit to the measured force profile of a sample fibreglass leaf spring.

Figure 5.8: Photograph of a non-tapered patch of fibreglass leaf springs, during stress-strain experiments in a Tensile Strength Machine. The data obtained from these experiments are used for building a realistic model of the springs.

spring measured in a Tensile Testing Machine (see Figure 5.8). These experiments included cyclic compression and decompression as well as compression until failure. In the former we found out that the spring's yielding point is approximately at 30% of their rest length. For this reason we added an extra 5% for safety and place the end stops at 25% of their lengths to prevent them from over compressing.

To give an idea of how these parameters work, Figure 5.7 (left) shows a family of force profile curves in which both the rest length and the compression force parameters have been normalised to 1, and the arc angle varies from 0.5 to 2 rad. Varying the rest length scales the curves in the x direction; varying the compression force parameter scales them in the y direction; and varying the arc angle alters the regressiveness of the force profile (the degree to which the stiffness decreases with increasing compression). Skippy's springs need to be regressive for reasons explained below. To give an idea of how accurately this model fits the behaviour of real fibreglass springs, Figure 5.7 (right) shows the model curve underneath measured data from a test sample of real springs. Observe that the model reproduces accurately both the nominal force profile and the force hysteresis of the real spring.

The choice of fibreglass leaf springs was driven by the following criteria: (1) we require a high elastic energy to weight ratio, (2) spring behaviour must be highly repeatable, and (3) the force profile must be regressive. Item 1 rules out materials like spring steel, while item 2 rules out rubber. We need repeatability because one of the intended uses of Skippy is to study how well it can learn difficult acrobatic manoeuvres by physical practice (i.e., the same way that a human acrobat would learn them). Regressiveness offers two advantages over a linear or progressive spring: greater stored elastic energy

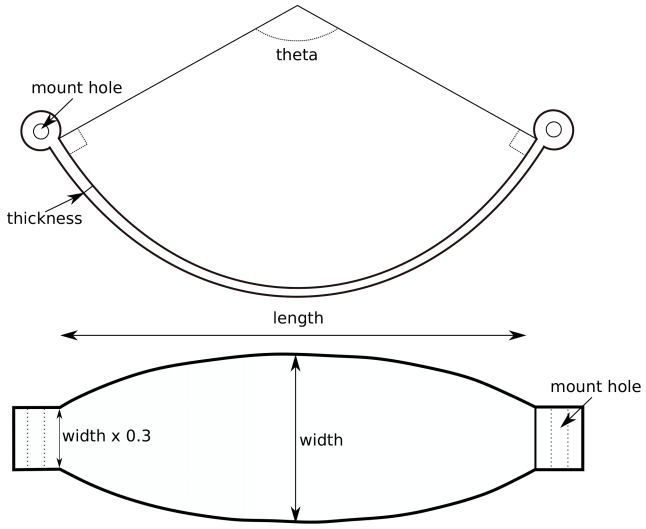

Figure 5.9: Geometry of the tapered fibreglass leaf spring. Top: side view; bottom: top view. The circular end points in the top figure, are the corresponding rectangular points in the bottom figure, which represent attachment points.

Name	Description	Value	LB	UB
Ankle spring parameters				
as_L	ankle spring rest length [m]	$\sim$	0.14	0.26
as_Phi	ankle spring arc angle [rad]	$\sim$	0.5	2
as_F25	force to compress ankle spring 25% [N]	$\sim$	950	1250
as_fh	force hysteresis of ankle spring	5%		
aslv_len	ankle spring lever length [m]	0.04		
aslv_ang	foot-to-ankle-lever angle at rest [rad]	2.1525		
q5r	ankle joint rest angle [rad]	-0.3		
as_alpha	ankle spring tapering parameter	0.3%		
Main spring parameters				
ms_L	main spring rest length [m]	$\sim$	0.14	0.26
ms_Phi	main spring arc angle [rad]	$\sim$	0.5	2
ms_F25	force to compress main spring 25% [N]	$\sim$	950	1250
ms_fh	force hysteresis of main spring	5%		
ms_alpha	mainspring tapering parameter	0.3%		

Table 5.6: List of leaf spring model and related parameters. Parameters with tilda in their 'Value' field do not have a fixed value and are optimised. LB and UB stand for lower and upper bound in the optimisation studies.

E-glass property	minimum value	maximum value
density [Mg m^{-3}]	2.55	2.6
bulk modulus [GPa]	43	50
Young's modulus [GPa]	72	85
Poisson's ratio	0.21	0.23
compressive strength [MPa]	4000	5000
tensile strength [MPa]	1950	2050
elastic limit [MPa]	2750	2875

Table 5.7: List of relevant properties of the E-glass used for making the fibreglass springs.

	Ankle end stop	
q5es	ankle end stop engagement angle [rad]	0.8858
aes_sc	ankle end stop specified compression [m]	0.05
aes_tsc	ankle end stop torque at spec comp [Nm]	12
aes_rtc	ankle end stop recovery time [s]	0.02
	Hip end stop	
q67es	hip end stop engagement position [rad]	0.14
hes_sc	hip end stop specified compression [m]	0.18
hes_tsc	hip end stop torque at specified comp [Nm]	100
hes_rtc	hip end stop recovery time constant [s]	0.007

Table 5.8: List of end stop parameters.

for a given stroke and maximum force, and greater stiffness at low force levels. Thus, regressive springs give Skippy enough energy to hop high, while still being stiff enough at low force levels to let Skippy balance quickly and accurately.

Figure 5.9 shows the shape of the spring. The tapered leaf spring is modelled as a strip of constant thickness and varying width that is wrapped around a (large diameter) cylinder so as to have a curved shape in its rest state. The tapering is such that the ends are 30% as wide as the middle. The spring's force is obtained by modelling the spring as a chain of 21 equal-length segments connected by linear torsional springs having stiffnesses propositional to segment width. A spline curve is then fitted to the data from this model, and the spline curve is used in the simulations. Finally, the springs are made from E-type glass. A few relevant properties of the material are presented in Table 5.7.

End Stops

Skippy has a hip and an ankle end-stop. Their purpose is to: (1) prevent over-compression of the ankle and hip springs (so they do not go beyond their yielding points), and (2) prevent self-collision. They are modelled as nonlinear viscoelastic progressive springs and dampers by Equation 5.1 and the parameters in Table 5.8).

Parameters 'q5es' and 'q67es' represent the engagement position for the ankle and hip end stop, respectively, and they are the result of preliminary optimisation studies that are not presented in this thesis. Skippy is expected to hit its end-stops during operation; thus, their activation points and behaviours must be taken into account. Based on the compression and the compression rate of the

Name	Description	Value
Vmax	battery supply voltage [V]	31
Imax	motor driver current limit [A]	45
IMUsat	IMU saturation limit [g]	16

Table 5.9: List of power supply and sensing limitations.

end-stop, an output force or torque is produced. The output is calculated according to the following formula:

$$F = Kx^2 + Dx\dot{x}. \tag{5.1}$$

F is the output torque or force, x is the displacement in metres or radians, and $\dot{x}$ is the compression rate. The gains K and D are calculated as follows:

$$\begin{aligned} K &= F_s d^{-2} \\ D &= 2t_c F_s d^{-2}. \end{aligned} \tag{5.2}$$

Parameters d, F_s and t_c are the end stop's specified compression, the torque at specified compression, and recovery time constant respectively (see Table 5.8).

Power Supply and Sensing

Skippy will have an 8-cell LiPo battery that has a nominal voltage of 29.6 V but a measured voltage of 31.0 V, which powers the motors and all electronics. To control the Maxon DCX32L [70] brushed motor the Pololu G2 High-Power Motor Driver 24v21 [81] is used, which has an instantaneous current limit of 50 A. In the simulations this limit is set at 45 A, because it is not a significant limiting factor on the robot's performance.

In hopping large shocks occur either during normal operation or crash landings, which can cause saturation of the IMU's accelerometer. The saturation limit of the Vectornav VN-100 [106] is 16 g. A result of saturation is that it confuses the IMU's Kalman filter, that is used for state estimation. To avoid this during normal operation a behavioural constraint is imposed as a limit to the maximum vertical ground reaction force during a behaviour (see Section 5.3.2).

Hardware Limitations and Limits

As mentioned in Section 4.2.2, it is reasonable to understand the hardware limitations and define them together with the models. Awareness of the robot's limitations can push behaviours towards the hardware's maximum potential in a safe way. To be used by the optimiser, these limits are converted to optimisation constraints. The constraints are all enforced, either directly in the dynamics simulation or as explicit constraints in the optimisation process, and they are the following.

1. **Voltage Limit**—set at 31 V based on the chosen batteries.

2. **Current Limit**—set at 45 A, maximum allowed current of the Pololu G2 High-Power Motor Driver 24v21 motor driver [81] is 50 A. Current is adjusted by the simulation model of the motor by reducing the commanded input voltage to the motor.

3. **Ring Screw Speed Limit**—set at 9,000 rpm. Even though the motor's limit is at 11,300 rpm, this limit is set to avoid resonance of the screw rod (Section 5.3.1).

4. **IMU Saturation Limit**—the normal component of ground reaction force during the motion must be less than 'fymax' (see Table 5.10). This constraint rules out behaviours in which the robot hits one of its end stops too hard. Such behaviours risk saturating the accelerometer in the VN-100 IMU [106], which is 16 g.

5. **Thrust-bearings Force Limit**—the pulling force on the lever during the motion must be less than 'thrust' (see Table 5.5). This limit is imposed by the thrust bearings holding the ring screw rod.

6. **Kinematic Constraints**

   - **Nut Position Limit**—the nut position variable (q_{10}) must be less than variable 'stroke' in Table 5.5 at all times during a behaviour. This constraint enforces a mechanical limit in the robot's mechanism.
   - **Leg and Hip Opening Limit**—the leg opening angle ($q_6 + q_7$, see Figure 5.4) must be less than 'q67max' (see Table 5.2) at lift-off. This constraint rules out behaviours in which the robot opens out too much.

These constraints define the operational range of the robot, or in other words the limits of physically valid behaviours, and they are all modelled as inequality constraints and apply for the entire behaviour. Constraint number 1 is imposed as bound on the input voltage variables. Constraints 2–3 are implemented in the simulation via the dynamics of the system. Finally constraints 4–6 are handled by the optimiser. They can be violated during a behaviour, and it is the optimiser's task to find valid behaviours that respect them.

5.3.2 Behaviour

This study is concerned only with Skippy's behaviour during its stance phase, which starts from the moment the foot touches the ground until the moment it lifts off. This is sufficient for our purposes because none of the performance objectives are concerned with Skippy's behaviour during its flight phase, which will be the task of a flight controller, that is not yet designed. The simulation starts after a plastic collision of the foot with the ground, and thereafter a feed-forward signal is applied to the motor until the foot lifts off the ground.

The set of parameters that describe Skippy's behaviour is listed in Table 5.10. As proposed in Section 4.2.3, these parameters can be classified into three groups:

Name	Description	Value	LB	UB
	Initial State Parameters			
$q4_i$	initial foot angle (measured from vertical)	$\sim$	-0.6	-0.05
$q10_i$	initial nut position (0=hip fully folded)	$\sim$	0.01	0.08
vcm0x	x coord CoM velocity before landing	*		
vcm0y	y coord CoM velocity before landing	*		
hcm0	angular momentum at CoM before landing	0		
	Outcome Parameters			
vcm1x	x coord CoM velocity at lift-off	*		
vcm1y	y coord CoM velocity at lift-off	*		
hcm1	angular momentum at CoM at lift-off	*		
fxmax	max horizontal GRF at lift-off	15		
fymax	max vertical GRF during motion	350		
q4min	min foot angle during motion	-1.4		
q67max	max hip angle at lift-off	2.2		
p1hmin	min height of P1 during motion	0.07		
	Input Parameters			
t1	duration of first Vin ramp	$\sim$	0.01	0.08
t2	duration of second Vin ramp	$\sim$	0.01	0.08
t3	duration of third Vin ramp	$\sim$	0.01	0.08
t4	duration of fourth Vin ramp	$\sim$	0.01	0.08
t5	duration of fifth Vin ramp	$\sim$	0.01	0.08
t6	duration of sixth Vin ramp	$\sim$	0.01	0.08
t7	duration of final Vin ramp	$\sim$	0.01	0.1
V1	relative voltage at time t1	$\sim$	-1	1
V2	relative voltage at time t1+t2	$\sim$	-1	1
V3	relative voltage at time t1+t2+t3	$\sim$	-1	1
V4	relative voltage at time t1+...+t4	$\sim$	-1	1
V5	relative voltage at time t1+...+t5	$\sim$	-1	1
V6	relative voltage at time t1+...+t6	$\sim$	-1	1

Table 5.10: List of 26 behaviour parameters that define Skippy's behaviour. Fields LB (lower bound) and UB (upper bound) are empty for constants. Time is in seconds, lengths in metres and angles in radians. Relative voltages are multiplied by Vmax (in Table 5.9) to obtain actual values. Initial and final conditions of the CoM (with '*' in their Value field) depend on the performance objective.

- initial parameters, which describe the state of the robot at the beginning of the stance phase;
- input parameters, which describe the voltage signal to the motor during the course of the stance phase; and
- outcome parameters, which describe things that must be true at the end of the stance phase, or which must be true during the whole of the stance phase.

The initial parameters describe certain key position variables and the velocity state of the robot just before the foot hits the ground. These parameters, together with a set of assumptions about the initial state of the robot, are sufficient to define the initial value of every state variable in the simulation (see Section 5.4.1 for details).

The input parameters describe the actions that the robot can take during a behaviour. They are the controlled variables and can be positions, torques, volts or any signal we wish to control. Because a

realistic model of a brushed DC motor is used, the input is a voltage signal, which is modelled as a sequence of seven ramps. After experimentation with fewer ramps, we found that six ramps provided substantially better results than five, but seven provided only a modest improvement over six. So we stopped at seven ramps on the assumption that we had reached the point of diminishing returns. Each voltage profile starts and ends with zero volts and is parametrized with six voltage values in the range of [-31 V, 31 V], and seven ramp durations restricted to the range of [0.01 s, 0.08 s].

The initial state and input parameters are sent to the simulator, where they are used to set up the simulation; but the outcome parameters are handled by the optimiser. Each of these parameters defines either a constraint on the actual outcome of a simulation run, or else an objective; and their roles vary from one performance test to the next. In all cases, they refer either to something that must be true at the end of the simulation, or to something that must be true throughout the simulation. For example, in Table 5.10 the parameter 'vcm1y' specifies the vertical component of centre-of-mass (CoM) velocity at the end of the simulation, which determines the height of the subsequent hop; whereas 'fymax' sets an upper limit to the vertical ground-reaction force at all times during the simulation. The purpose of this parameter is to discourage behaviours that rely on hitting an end stop hard, which risks saturating the IMU's accelerometer. Another parameter of this kind is 'p1hmin' (Table 5.10), which disallows behaviours in which the robot's hip hits the ground.

In every performance test, all of the input parameters and the two initial parameters 'q4i' and 'q10' (which define the configuration of the robot at landing) are treated as independent optimisation variables, and the rest are fixed. Where a specific value is given for a fixed parameter in the table, it means that the parameter value is the same in every test. Where no value is given, it means that the value depends on the test.

Behaviour Constraints

Behaviour constrains define how the robot should achieve its objectives as well as what is considered an acceptable outcome for a behaviour (Section 4.2.3).

1. **Slip Constraint**—the magnitude of the tangential component of ground reaction force at lift-off must be less than 'fxmax'. This constraint rules out behaviours which, if executed by a real robot, would cause too much slipping of the foot during the last few milliseconds before lift-off. We decided to not use a friction cone constraint, because it will almost always be violated in the final milliseconds before lift-off.

2. **Behaviour Time Limit**—the sum of the first six voltage ramp durations must be less than 0.3 s.

3. **Ramp Effectiveness**—the last ramp must start at least 10 ms before lift-off. Its purpose was to ensure that all seven ramps had an effect on the behaviour. This constraint was not active in the experiments presented in Section 6.6.

4. **4-bar and Ground Collision Limit**—the computed height of the point P1 in the mechanism (see Figure 5.4) must be greater than 'p1hmin' (Table 5.10) at all times during the motion. This constraint rules out behaviours in which the bottom of the 4-bar linkage hits the ground.

5. **Foot and Ground Collision Limit**—the foot angle during the motion must be greater than 'q4min' (see Table 5.10). This constraint rules out behaviours in which the robot's ankle touches the ground.

6. **Performance Objective Constraints**

   - **Angular Momentum**—the angular momentum of the robot about its CoM at lift-off must equal 'hcm1'.
   - **Vertical CoM Velocity**—the vertical component of CoM velocity at lift-off must equal 'vcm1y'.
   - **Horizontal CoM Velocity**—the horizontal component of CoM velocity at lift-off must equal 'vcm1x'.

Constraints 1–5 are translated into inequality constraints and are all handled by the optimisation algorithm. This means that any value below these limits produces a valid behaviour. For example, any behaviour that lasts less that 0.3 s is acceptable. The 'Performance Objective Constraints' are modelled as equality constraints. Depending on the performance objective, these constraints might not apply or be allowed some slack. For example, in vertical hops the robot must have zero angular momentum. However, this is a very strict constraint and a small relaxation (such as allowing an angular momentum error of $\pm 0.2 \text{kgm}^2\text{s}^{-1}$) leads to more useful results, without a significant loss of performance or violation of the design objectives.

5.3.3 Assumptions

In this section the assumptions in the modelling of Skippy's mechanism and its behaviours are presented. As explained in Section 4.2.1 assumptions are an important part of the modelling process and can lead to simplified and computationally cheaper to execute models. The assumptions taken in modelling of Skippy and its behaviours are the following.

1. 2 D dynamics—although Skippy will be a robot able to balance and hop fully autonomously in 3 D, the design study in this thesis considers only behaviours in a plane (the robot's sagittal plane). The justification for this simplification is that hopping is mostly a planar activity, and a planar study is sufficient to capture all of the main energy flows during it. Even after this simplification, the robot still has almost 80 design parameters, of which only a few of the most influential were chosen for optimisation. Specifically, in the first study presented in Section 6.5, eight design parameters were optimised, and in the final study of Section 6.6 seven parameters were varied. These parameters were selected according to the results of preliminary experiments and a sensitivity analysis (see Section 6.6.2).

2. Initial state-parameters.

    - It is assumed that both the main spring and the ankle spring are at their rest lengths before the robot lands. This implies that the ankle joint is at its rest angle.
    - It is assumed that all of the robot's joint velocities are zero before the landing, but the robot as a whole is moving as described by the initial-value parameters 'vcm0x', 'vcm0y' and 'hcm0' (see Table 5.10).

3. A perfectly plastic collision without slipping occurs when the foot touches the ground.

4. The robot is constrained to not slip during the stance phase. In reality, Skippy will slip; however, excessive slipping is prevented with constraint 1 in Section 5.3.2.

5. The environment is horizontal, flat and empty.

6. Motor temperature—at the beginning of each stance phase is set to 25°C.

7. A flight-phase controller that can bring Skippy into the desired initial condition for the next hop can be designed.

5.3.4 Conclusion

Tables 5.2 – 5.10 list the complete set of the 78 parameters that describe Skippy's hardware and limitations, and 26 parameters for its behaviour. To understand the kinematic parameters, see Figure 5.4. Observe that in addition to the usual kinematic and inertia parameters that one would expect to see in any dynamic model of a robot, these tables also list spring and end stop parameters, efficiencies of important parts (i.e., ring screw transmission mechanism), as well as limits on motor voltage, current and speed, and ring screw thrust force.

Even for a relatively simple robot such as Skippy, more than a hundred parameters are required to obtain a realistic model. However, not all of these parameters need to be optimised. A tool that can help understand which parameters or combinations of them have the highest effect on the robot's performance is Sensitivity analysis (Section 4.5.2). Alternatively, independent optimisation tests can be performed by the designer to determine the most important parameters. Finally, some of these parameters are dependent on early design decisions. For example, we have decided on the motor and IMU to be used in Skippy prior to this design optimisation study, and hence these parameters are excluded from the optimisation.

5.4 Selecting Software: Simulation

In this section the software used for the modelling and optimisation study is presented. Details about the simulation model implementation and its architecture (top-level layer and sub-layers) are presented and discussed.

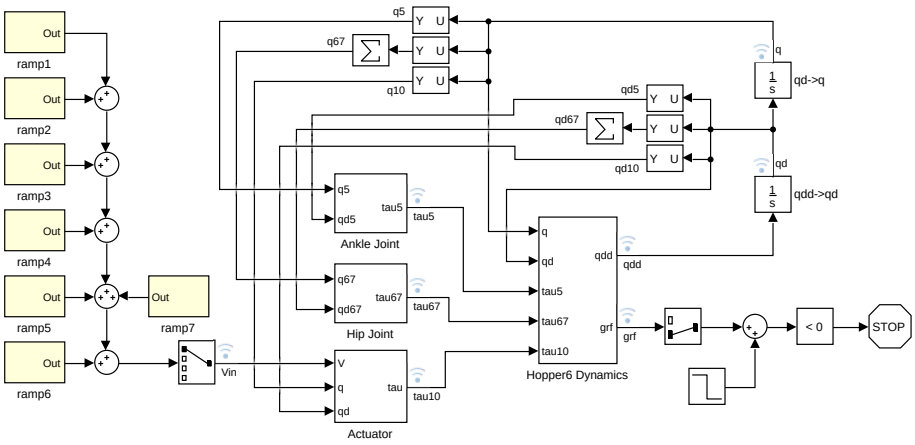

Figure 5.10: Top-level block diagram of the Simulink model of Skippy.

5.4.1 Models and Blocks

For simulation and modelling we chose *Simulink* [69] from *Mathworks*, which is a widely used software with a free licence for academia. It provides a graphical workflow which facilitates the modelling process. Moreover, it comes with a variety of solvers for running simulations.

For Skippy a high degree of accuracy is required because this is an open-loop simulation of a robot that is making fast movements close to an unstable balanced configuration, so any deviation from the theoretically exact solution grows exponentially with time. An even higher accuracy is required when gradient-based methods are used because they perform a numerical differentiation of the outcome of the simulation with respect to the input and initial-value parameters. For this reason, the solver 'ode45' is used, which is a variable-step continuous solver. In addition, the parameter *relative tolerance* is set to 10^{-6} for global optimisation and 10^{-8} for gradient-based methods.

Figure 5.10 shows the top-level block diagram of the Simulink model of Skippy used in the design optimisation process. The major parts of this model are as follows.

- input signal formation (yellow blocks);
- ankle joint subsystem, which implements the ankle spring and end stop;
- hip joint subsystem, which implements the hip end stop;
- actuator subsystem, which implements the motor, ring screw and main spring;
- dynamics subsystem, which implements the planar rigid-body dynamics of the robot mechanism; and the
- stopping criterion, which stops the simulation when the lift-off condition is detected.

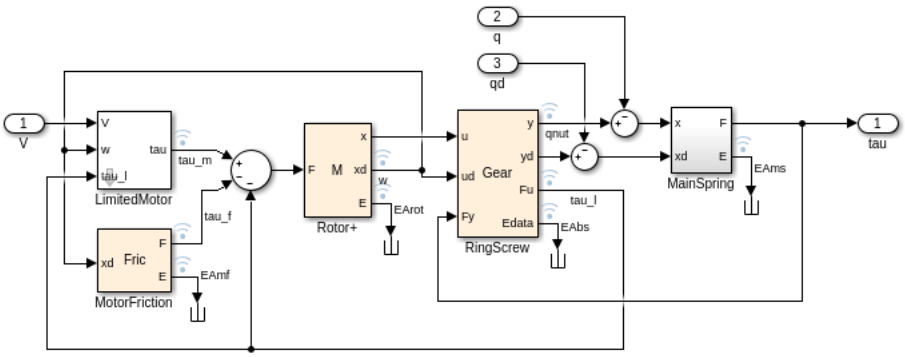

Figure 5.11: Actuator sub-system of the Simulink model of Skippy.

Initial Conditions

As mentioned earlier, the initial-condition parameters must be combined with some assumptions in order to fully determine the initial state of a simulation. These assumptions are described in Section 5.3.3. It is assumed that a perfectly plastic collision without slipping occurs when the foot touches the ground; and an impulsive dynamics calculation based on this assumption is used to calculate the initial values of all of the velocity variables in the system, except the motor's rotor velocity, which is set to zero.

Termination Condition

The simulation terminates when the foot lifts off the ground. In the simulator this can be modelled as a condition where the vertical component of the ground reaction force becomes negative. As long as the foot is in contact with the ground this value must be positive.

One consequence of the assumption of a perfectly plastic collision is that the normal component of ground reaction force can be negative during the first one or two milliseconds of the stance phase. For this reason, the lift-off detector is disabled during the first few milliseconds of a simulation run so as to avoid a premature termination. This is the purpose of the step block in Figure 5.4.

Actuator System

The actuator subsystem implements the electric motor, the ring screw and the main spring models. This subsystem takes into account the rotational inertias of the motor's rotor and the ring screw rod when calculating the angular acceleration of the rotor in response to motor torque and load torque. However, as this is an out-of-plane motion, the angular momentum due to this motion is not taken into account when calculating the motion of the robot. This is theoretically correct if one assumes that the robot is physically constrained to move in a vertical plane.

The input voltage is fed to the 'LimitedMotor' block, which implements the DCX32L [70] model. From the output torque of the motor, a viscous friction force is subtracted, implemented by the block

'MotorFriction'. The resulting torque accelerates a mass with an inertia equal to the sum of the rotor, the ring screw rod, and ring screw nut (see Table 5.4 and 5.5). The rotation causes the ring screw nut to translate and compress the main spring, which is attached to it. This produces the final output of the system, which is the force that is applied to the lever.

MotorFriction—the viscous friction is calculated as follows:

$$F = A \tanh(b\,xd), \tag{5.3}$$

with $b = 1$ and $A = K_\tau I_{nl}$ (see Table 5.4), which is the friction magnitude, and xd is the angular velocity of the rotor.

Motor System

The 'LimitedMotor' system contains the implementation of the DC motor model, its thermal model, the current saturation and ring screw speed limits. The system's inputs are: (1) the commanded voltage, (2) the angular velocity of the rotor, and (3) the input torque of the ring screw, which is the motor's load torque. The system's output is the torque tau that is the torque generated by the magnetic field. The friction torque and the load torque are then subtracted from this, and what remains is the torque that accelerates the rotor.

This block contains the implementation of the thermal model and the circuit model for a DC motor. The motor's circuit is modelled either as an LR or just R circuit. Although the block is equipped to simulate an LR circuit, we never use it because it makes the simulations more expensive. All simulations use the assumption $L = 0$. Specifically, using Kirchoff's Voltage law the terminal relationship for the LR circuit is given by

$$V = V_{\text{emf}} + IR. \tag{5.4}$$

V_{emf} is the back EMF voltage that is calculated by the following equation:

$$V_{\text{emf}} = K_\tau\,\omega. \tag{5.5}$$

The parameters for these equations correspond to the ones presented in Table 5.4, and ω is the angular velocity of the rotor. This block solves Equation 5.4 for the current value (I), and calculates the output torque tau as follows:

$$tau = K_\tau I. \tag{5.6}$$

The thermal model uses the thermal parameters of Table 5.4, the electric losses:

$$P_{loss} = I^2 R, \tag{5.7}$$

and the losses due to motor friction:

$$F_{loss} = |\omega| K_\tau I_{nl}, \tag{5.8}$$

to calculate the temperature of the coils and the motor, as well as the varying value of the resistance R. Due to the flow of current, the temperature of the winding increases. Metal conductors have a positive temperature coefficient so as the winding temperature increases, its resistance also increases.

Dynamics System

The dynamics subsystem implements a closed-loop hybrid dynamics calculation in planar-vector arithmetic [29] using functions in the library *Spatial_v2* [34]. In this calculation, joints 1 and 2, which are prismatic joints in the x and y directions, are treated as inverse-dynamics joints, and are held at zero throughout the simulation. This is equivalent to enforcing a rolling contact without slipping between the foot and the ground regardless of the value of the ground-reaction force. The outputs are the accelerations of the forward-dynamics joints and the forces required to maintain the rolling contact. These forces are interpreted as ground-reaction forces, even if the normal force is negative or the tangential force lies outside the friction cone. However, the stopping criterion does monitor the normal force, and stops the simulation as soon as it goes negative; and the slip constraint mentioned in Section 5.3.2 rules out behaviours in which the tangential force lies outside the friction cone by more than a tolerance threshold value specified by 'fmax' (see Table 5.10).

Hip Joint System

This sub-system contains only an end-stop block, which is described in Section 5.3.1. An output is produced only when $q_6 + q_7 < q67es$ (see Table 5.8). If this condition arises then the hip end-stop is engaged and a torque opposite to the hip and leg closing motion is generated as explained in Section 5.3.1.

Ankle Joint System

The ankle joint subsystem contains: (1) a block implementing the ankle spring, and (2) an end-stop block that is activated when $q_5 < q5es$ (see Table 5.8).

The Simulink sub-system of the ankle joint is presented in Figure 5.12. The inputs to the system are: (1) the ankle joint's angle (q_5), and (2) its angular velocity ($\dot{q_5}$), and the output is the torque *tau* about the joint axis. Value q_5 is subtracted from the joint rest angle (asH is parameter 'aslv_ang' in Table 5.6) to obtain the derivation of the joint's angle from rest, which is used for computing the compression of the ankle spring. The ankle spring system consists of two main blocks: the (1)

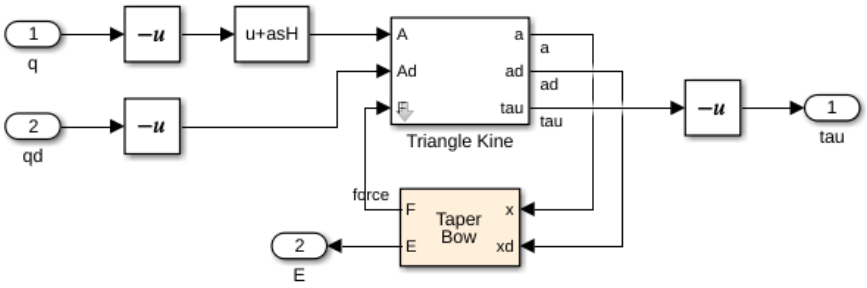

Figure 5.12: Ankle spring sub-system of the Simulink model of Skippy.

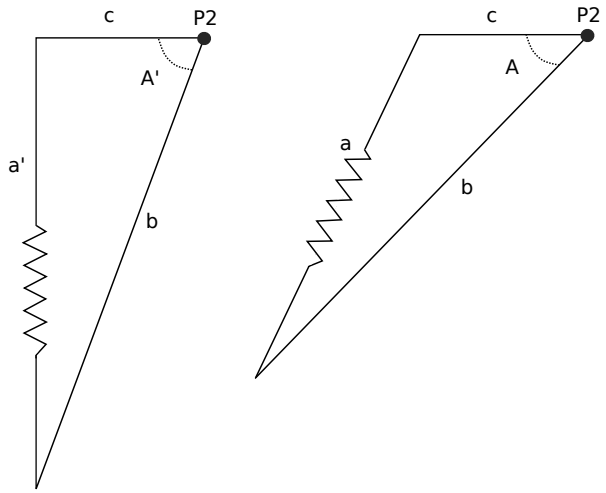

Figure 5.13: Triangle defined by the ankle spring's attachment points and ankle joint. The left figure shows the configuration at its rest position, and the right figure shows the same spring after it has been compressed by $(a' - a)$ m. Drawing is not to scale. See bottom right Figure 5.4 for reference.

'Taper Bow' block, which implements the leaf spring model, and (2) the 'Triangle Kine' block. The ankle spring is a compression spring that is attached to the foot and the leg. The leaf spring model is implemented by a Matlab script, and is described in Section 5.3.1

The spring's attachment points and the joint axis form a triangle (see bottom right Figure 5.4 and Figure 5.13 for a simplified view). To obtain a torque about the ankle axis some simple calculations are required that are performed inside the 'Triangle kine' block. This block transforms the joint angular position and velocity to the corresponding linear quantities that are fed to the ankle spring, and outputs the torque about the ankle joint axis.

Figure 5.13 shows an abstract representation to demonstrate the ankle spring configuration with the ankle joint. Parameter c is equal to parameter 'aslv_len' in Table 5.6, parameter a' is the ankle spring's rest length (parameter 'as_L'), and parameter A' is the ankle joint rest angle (parameter 'aslv_ang'). The spring's attachment point on the leg has a distance of c m from point P2 which coincides with the rotation axis of the ankle joint. The distance b from P2 to the spring's attachment point on the foot is constant and can be calculated by the following equation:

$$b = c\cos A' + \sqrt{a'^2 - c^2 \sin A'^2}\,. \tag{5.9}$$

Given an angle of the ankle joint A the spring's new length a can then be calculated by the cosine rule

$$a = \sqrt{b^2 + c^2 - 2bc\cos A}\,. \tag{5.10}$$

Following this, the spring's rate of compression is the derivative of Equation 5.10, which is:

$$\dot{a} = bc\sin A\, a^{-1}\,. \tag{5.11}$$

Then the compression and the compression rate of the spring are fed to the 'Taper Bow' block which produces a linear force F as output. Finally, the output torque tau at the ankle joint resulting from force F acting along a is

$$tau = \frac{F\dot{a}}{\dot{A}}, \tag{5.12}$$

and is added to the output of the end-stop to produce the torque about the ankle joint axis.

Energy Flows

The simulation takes into account all energy flows, including all energy losses. In the presented Simulink diagrams these values come from signals having names starting with the letter E. This is performed to allow an *energy audit* on the results. An energy audit is a post-processing technique for detecting energy discrepancies that occurred during the course of a simulation. At each time value in the logged time data, it compares the total energy in the system at that time with the energy that was present at the start of the simulation, plus all of the new energy that has been added so far (from the battery pack), minus all energy losses since the beginning of the simulation. The two numbers should be the same. The energy audits are used for three purposes:

1. debugging,
2. checking the accuracy of the simulation, and
3. to study the energy flows to help us understand what is happening during the stance phase.

Energy flows of specific behaviours will be presented in the next chapter.

5.5 Balancing Studies

The purpose of this section is to present and justify the use of the *velocity gain* (VG) as a balancing performance measure for Skippy. The VG is described in Featherstone [30]. This measure provides means of quantifying a robot's ability to balance and can be obtained from a single calculation. It is expressed as a ratio of change in the robot's state of motion to the amount of effort that the actuator needs to achieve that change. A higher value of the VG means less work is needed for the robot to change its state and balance.

5.5.1 Experimental Results

To prove the effectiveness of the balancing algorithm that will be used in Skippy and gain insights on its design, we built its precursor, a robot called Tippy, which is depicted in Figure 5.14. This robot is similar to Skippy. It is a 2 DoF actuated 3 D balancer with similar mass and characteristics. It has a leg, a torso and a crossbar; however, unlike Skippy, Tippy has a fixed base, and its foot is connected to it via a spherical joint. This robot was created for the sole purpose of balancing. In addition, it does not have a 4-bar linkage, but a Harmonic Drive [47], and a single revolute joint at the hip.

With Tippy, the first experimental results on Featherstone's [31] balancing algorithm were presented in the work of Driessen et al. [24]. For this experiment, Tippy was constrained to rotate only in the roll plane, by having its hip joint locked and by actuating only the crossbar. In the balance control literature this configuration is referred to as a *Reaction Wheel Pendulum*. The practical results presented in this work demonstrated satisfactory high-performance balancing and tracking; however, the robot's performance was impeded by its body's low stiffness. The results make evident that in order to obtain a high-performance balancing machine a stiff body is required. Having obtained this information we are trying to build Skippy to be as stiff as possible, by employing stiffer materials and structures, and by replacing the elastic Harmonic Drives [47] used in Tippy (which we found to be a source of instability with high controller gains).

The velocity gain has also been proven to work experimentally in the work of Gonzalez et al. [39]. In this paper the same balancing algorithm that was used in Tippy, and which will be used in Skippy, is applied to the HyQ quadruped robot to perform line walking. In Featherstone [30] a linear velocity gain (LVG) and an angular velocity gain (AVG) are described. For this thesis and the works referenced the AVG is used.

5.5.2 Angular Velocity Gain

In Skippy, the angular velocity gain is used, as described in [30], which is defined as the step change in the angular velocity of the robot's centre of mass about the support point divided by the step change in the velocity of the actuated joint that causes it. Skippy balances in the sagittal plane using its hip joint, which has a limited range of motion. Therefore, the larger the magnitude of the velocity gain, the larger the balance error that can be corrected by the hip joint before it reaches an end stop. So

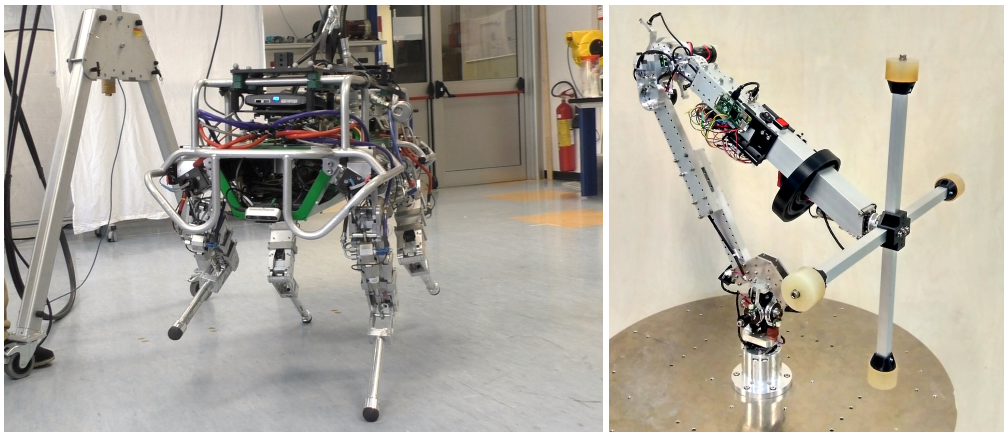

Figure 5.14: Left: HYQ balancing on a line using only two feet [39]. Right: Tippy a 3 D balancing robot [24].

the performance objective for balancing is simply to maximise the magnitude of Skippy's angular velocity gain.

Because the AVG is dependent on the state of the robot, for the optimisation study it is measured in a standard configuration (torso horizontal, robot balanced). Thereupon, an impulse is applied on the actuated joint and the AVG is measured by the following calculation

$$G_{\omega} = \frac{\Delta\dot{\phi}}{\Delta\dot{q}_a}. \tag{5.13}$$

Parameter $\Delta\dot{\phi}$ is the change in Skippy's CoM angular velocity, and $\Delta\dot{q}_a$ is the change of the angular velocity of the actuated joint as a result of the applied impulse. The resulting value G_{ω} is the performance objective reflecting the robot's balancing ability that is used in the optimisation experiments, which we wish to maximise.

In summary, the AVG has proven to be a robust and effective measure of a robot's balancing ability in two cases: (1) its precursor Tippy, which is a similar robot to Skippy, and (2) the more complicated and significantly heavier quadruped robot (90 kg) HyQ.

5.6 Conclusion

In this chapter a monopedal balancing and hopping robot named Skippy was presented. The aim of this chapter is to define the robot's mechanism and behaviour models, together with the chosen simulation environment and example model implementations. This chapter follows the philosophy of the proposed design optimisation approach described in Section 4.2, which is based on creating realistic models of the robot, while trying to keep these models as simple as possible.

First a literature review on hopping and jumping robots was presented, together with a critical evaluation on their design purpose, process and choices. This section aimed to locate Skippy in the current literature, discuss challenges of hopping, the state of the art, and compare design decisions of other hopping robots with Skippy. Following that, Skippy's components and limitations were presented, together with some important design decisions. Then, the behaviour of the robot was modelled and explained.

Next, models of various parts of Skippy were presented in Simulink, which is the software that was chosen for modelling and simulation. Following that, the velocity gain was introduced, which is a measure of a mechanism's balancing ability that is used as a performance objective in the optimisation studies. Finally, some real applications of this measure were presented to prove its effectiveness and justify its use.

In the next chapter the optimisation framework and process are applied for the design and behaviour co-optimisation of Skippy. This happens as a two-step design study, where the results of the first study are used as a seed for the second and more advanced study.

Chapter 6

A Versatile Hopper

6.1 Introduction

In the previous chapter a monopedal hopping and balancing robot named Skippy was introduced, and a model of its hardware and behaviour was presented. In this chapter, the proposed design optimisation approach is applied to Skippy for behaviour and design co-optimisation.

The challenging aspect of Skippy's design is that it is required to meet a large number of high-performance objectives, which are close to its maximum physical potential. Specifically, Skippy must be a very skilful vertical hopper, which means to hop as high as possible, close to its physical performance envelope, and at the same time be able to demonstrate a large repertoire of other demanding behaviours such as: (1) vertical hops, (2) somersaults, (3) travelling hops, and (4) having a high balancing ability. Some of the physical requirements of these behaviours conflict with each other, which means that optimising for one leads to a reduction of performance in another. For example, a high hop requires a significant amount of energy to be stored in one or more springs, whereas high-performance balancing requires the robot mechanism and actuators to be stiff. Furthermore, stiff springs are difficult to load from rest, due to the lack of momentum. Another challenge is that this robot will be built, which means that all the components of the design must be available off-the-self or be easily manufactured.

The optimisation experiments use the 2-layer optimisation framework and the optimisation methodology presented in Chapter 4. In the first layer of the framework a set of Skippy's design parameters are optimised, and in the second layer a set of performance objective scores are computed, which are either behavioural objectives or a physical property of the robot. The outcome of the second layer is: (1) a set of performance scores, (2) the optimal behaviours to achieve each objective score, and (3) the vector of constraint values. These results are passed to the first layer and are used to evaluate and generate new designs.

This study is performed in two steps; in the first part, the objective is to find a design that can meet a set of performance objectives, and the second part is a more advanced study based on the results of the first part. The summary of the experiments presented in Sections 6.5 and 6.6 is the following.

1. Discover a design that can: (1) perform a series of four increasing-height vertical hops up to a 3.2 m hop, (2) perform a series of four decreasing-height vertical hops down to a halt, (3) be a skilled balancing machine, and (4) perform travelling hops. The objective of this study is to discover at least one design capable of meeting all of the aforementioned performance objectives, something that did not exist and was difficult to achieve without the application of the proposed approach.

2. Given a design capable to meet the previously mentioned performance objectives, a new round of optimisation studies is performed by adding extra evaluation criteria and performance objectives, which are: a repeated vertical hop of 2 m, and a 2 m triple somersault. In addition to these, the model is updated with a more realistic leaf-spring model (see Section 5.3.1).

The result of the first experiment is a Pareto front of optimal designs, capable of achieving all of the desired performance objectives. From this set, the design with the best trade-off is selected based on a set of evaluation criteria. Thereupon, the selected design is used as a seed for the second design study. In this study the complete proposed optimisation process is presented. In addition, more sophisticated evaluation criteria are introduced for selecting the best design, and a comparison is made between the final result of the first experiment and a small set of selected designs from the second experiment. Finally, the optimal behaviours of one selected design are presented and analysed.

6.2 Optimisation Constraints

The optimisation constraints are the same as presented in Sections 5.3.1 and 5.3.2, and are summarised below.

- Inequality Constraints
    1. IMU Saturation Limit.
    2. Thrust-bearings Force Limit.
    3. Nut Position Limit.
    4. Leg and Hip Opening Limit.
    5. Slip Constraint.
    6. Behaviour Time Limit.
    7. Ramp Effectiveness.
    8. 4-bar and Ground Collision Limit.
    9. Foot and Ground Collision Limit.
- Equality constraints
    1. Angular Momentum.

2. Vertical CoM Velocity.
3. Horizontal CoM Velocity.

The constraints (1) 'Voltage Limit', (2) 'Current Limit', and (3) 'ring screw speed limit' are directly applied in the simulation. As explained in Section 6.6.9, these values are an indication of the robot's effort, which also reflects (together with the constraints) how close the mechanism is to its true physical potential. In this thesis, they are used only as a post-processing evaluation criterion; however, these values could provide beneficial information to the optimiser and can have an active role in the optimisation process as extra objectives or constraints.

Scaling—some of these constraints compare large numbers, such as thrust forces in newtons, while others compare small numbers, such as voltage ramp durations in seconds. So scaling factors are applied to each individual constraint so that the numbers being compared are close to 1.

6.3 Determining the Problem Type

To select appropriate methods to perform the optimisation experiments, the problem type must be determined (see Section 4.3). The examined optimisation problem falls into the following categories.

Continuous—the system consists of continuous models, and does not have discrete variables.

Constrained—by definition the problem has 12 constraints.

Multi objective—the aim is to design a versatile robot capable of meeting several performance objectives (more than 10 in each experiment).

Deterministic—the models do not include any stochastic elements (i.e., the system produces the same output for the same input).

Non-linear—the problem consists of non-linear dynamics and models (e.g., regressive springs in Figure 5.7), and non-linear relationships have been experimentally observed between the independent variables and the performance objectives.

Non-convex—in a series of preliminary experiments, multiple local solutions were discovered for a given objective. This means that the problem has more than one mode, and hence it should be addressed with a global optimisation algorithm (Section 2.2.3).

6.4 Selecting Software: Optimisation

Given the problem type and the available resources (i.e., licences for commercial software and available computational power) ModeFrontier [27] (MF) is a suitable choice as the optimisation software. MF offers a large selection of optimisation algorithms that fit the description of the examined problem, in addition to graphical work flows for the optimisation process, sophisticated DOE methods, and a variety of post-processing and visualisation tools for data analysis. In addition, the work flow can communicate with Matlab [67] (and Simulink [69]), which are used in this thesis to perform accurate and realistic dynamics simulations.

6.4.1 Optimisation Algorithm

For the optimisation experiments the *MOGA-II* [80] algorithm was chosen. MOGA-II is a proprietary version of the multi-objective genetic algorithm [18] for global optimisation that can efficiently handle problems that are multi-objective, non-linear and constrained. The overview of the algorithm is presented in Algorithm 1.

Starting with an initial population set as an input (parents), the algorithm generates new offspring using the classical genetic operators of: (1) mutation, (2) selection, and (3) cross-over. When a new generation is complete (i.e., the number of offspring is equal to the number of parents) the fitness of all the individuals is evaluated to determine the best solution based on the optimisation objectives and constraints. All the non-dominated designs (i.e., the Pareto front) are stored in the *elite set*. Then, a new parent set is generated from the elite set and the best designs of the previous generation. The process is repeated until the maximum number of generations is reached. The hyper-parameter values of the algorithm are presented in Table 6.1.

Genetic Encoding and Operators

The population consists of individuals, with each being a solution to the examined problem. The individuals, which can also be called chromosomes are described by the values of the independent variables (which are also called genes). To generate a new offspring, one of the genetic operators is applied based on the set of probabilities, as shown in Algorithm 1.

Mutation—given a mutation ratio, a percentage of the independent variables changes value. For example, if the mutation ratio is 50% and there are eight genes, then exactly four out of eight genes will be randomly selected and change value, such as the new value still remains within the given bounds. The probability that an individual is mutated is dependent on the hyper-parameter ‘P(mutation)’. A high mutation probability introduces randomness, and hence promotes exploration; however, if this value is not properly selected it may result in a poor convergence rate. A mutation ratio of 1 results in a random search.

Algorithm 1: MOGA-II Generational Evolution

```
Result: Feasible Designs or Behaviours
Input: Initial Population Set;
Evaluate Fitness of Initial Population Set;
while N. of Generations is Not Reached do
    while N. of New Offspring < Generation Size do
        x ∈ Uniform(0,1);
        if x < P(Directional Cross-Over) then
            Directional Cross-Over;
        else if x ≤ 1-P(Selection)-P(Mutation) then
            Classical Cross-Over;
        else if x ≤ 1-P(Mutation) then
            Selection;
        else
            Mutation;
        end
    end
    Generation Fitness Evaluation;
    Elitism Selection;
end
```

Selection—this operator is used to simulate the process of natural selection in nature. A solution is chosen and passes unaltered to the next generation for future breeding. The probability that a solution is selected is equal to the hyper-parameter 'P(selection)', and is dependent on its fitness. A higher fitness value means higher chances of a solution to be selected (for more details see Coello et al. [18]). This operator ensures that fit designs are maintained across generations.

Elitism—creates a set that contains the fittest candidates from all generations. When Elitism is switched on each new generation consists of the Elite set and the best offspring generated by the previous generation. The elite set is updated after the completion of a generation, and is always less or equal than the population size. This operator guarantees that a solution will not be degraded over the course of evolution and has been shown to improve the convergence of the algorithm [18, 80].

Cross-Over—this operator combines the genes of two individuals to generate new offspring. This operator promotes diversity in the population, while combining characteristics of the two parents. Two chromosomes randomly exchange genes to create new offspring. The probability a chromosome is selected is dependent on the hyper-parameter 'P(Cross-Over)'.

Directional Cross-over—this operator is similar to the cross-over operator; however, it works under the assumption that, based on the fitness of the parents, the direction of improvement can be estimated. This is done by a heuristic which determines which parents should generate new offspring. The probability of a chromosome to be selected is equal to the hyper-parameter 'P(Directional Cross-Over)'.

The algorithm decides which operator to apply by generating a pseudo-random value x between 0 and 1. By using the presented operators, and the absence of dependency on gradient information the optimiser has the ability to escape local optima. The choice of the hyper-parameter values (which are probabilities) determines the trade-off between exploration and exploitation of the algorithm. In the presented experiments, a combination of these hyper-parameter values was chosen to promote in equal terms exploration and exploitation; however, during initial experiments, where no known solution existed, a combination of hyper-parameters was selected that tilted the scale towards exploration. Finally, the values of these parameters should be decided based on the available current information and experiment purpose.

Constraint Handling

Constraint handling can effectively guide the optimisation by passing information to the optimiser about the margin of success and failure (Section 4.1). For handling the constraints, the algorithm offers two options:

1. penalising objectives, or
2. treating constraints as extra objectives.

Due to the large number of already existing objectives, the option of penalisation seems to be the most reasonable. In this case, for each individual, a constraint vector score is calculated, which is subtracted (or added, if the objective is minimised) from the value of the objective function. By using this option it is possible for unfeasible designs to have a higher score than feasible designs, which can help the optimiser to navigate towards regions with potentially better solutions. An additional variable for constraint handling that is utilised in this study is the 'tolerance value'. The purpose of this value is to inform the optimiser how severe is a violation of a constraint.

Tolerances can be applied to constraints that are allowed with an error margin, such as the Performance Constraints (see Section 5.3.2). For example, in vertical hops the robot is required to have zero angular momentum and zero vertical velocity. However, this constraint is very strict and even if it is met in simulation, it may lead to over-optimisation. For this reason, a small tolerance is allowed in these constraints to give the optimiser more freedom to explore the search space, and to avoid over optimisation of the theoretical model.

Algorithm Type

The software also offers three implementations of the algorithm. These are:

1. Generational Evolution—the hyper-parameters remain constant and the next generation is created only when all design evaluations are complete;
2. Steady Evolution—new designs are generated regardless of whether or not a generation is completed;

Parameter Name	1st Layer (Mechanism)	2nd Layer (Behaviour)
Initial population	DOE + User-defined	DOE + User-defined
Population Size	∼	∼
Number of Generations	∼	∼
Prob. of Selection	5%	5%
Prob. of Mutation	10%	10%
Prob. of Directional CO	50%	50%
DNA String Mutation Ratio	5%	5%
Elitism	Enabled	Enabled
Treat Constraints	Penalising Objectives	Penalising Objectives
Algorithm Type	Generational Evolution	Generational Evolution

Table 6.1: Hyper-parameters for the MOGA-II algorithm for the mechanism and the behaviour layers. Different DOE techniques are used to promote exploration of the search space by adding points that maximise the minimum distance between the existing points. 'User defined' are mechanisms that were found from previous experiments and are known to perform well. Initial population and population size differ for the two experiments. 'CO' stands for crossover.

3. Adaptive Evolution—a heuristic is used to tune the hyper-parameters of the algorithm.

For the presented experiments, the Generational Evolution implementation was chosen. The second option provides faster results since no resources are wasted while waiting for the generation to be completed; however, sub-optimal designs can propagate to the next generation, since evaluation is performed without full information. Regarding the third option, the selected hyper-parameter values of Table 6.1 were selected after a series of independent exploratory experiments, and hence there is no need to go with this option, which should be used when there is no clear idea on how these parameters should be set.

6.5 Hopping, Travelling and Balancing

In this section the proposed optimisation approach is applied to Skippy. The objective of the first out of the two studies is to reach the performance envelope in vertical hopping as well as to perform moderate-difficulty travelling hops, and balance as efficiently as possible.

To select appropriate values for these performance objectives, a series of single-objective preliminary optimisation experiments (which are not presented in this thesis) were performed, using the gradient-based algorithm SQP in Matlab's optimisation toolbox [68]. In these exploratory experiments the model of Skippy presented in Chapter 5 was used, but not the proposed optimisation approach. The behaviour and design parameters were simultaneously optimised, but for only one objective at a time (and not in two layers as in the proposed framework).

Obtaining a design capable of achieving a large set of performance objectives was very difficult with the aforementioned approach. For this reason a more sophisticated way was sought, which was the motivation to develop the proposed design and behaviour co-optimisation approach.

6.5.1 Performance Objectives

The objective of this study is to find a design that can: (1) perform a series of four increasing-height vertical hops (zero angular momentum and horizontal velocity) up to a 3.2 m (8 m/s) hop starting from rest, (2) perform a series of four decreasing-height vertical hops down to a halt, (3) be a skilled balancing machine, (4) start travelling following a vertical hop, keep travelling, and stop travelling and go back into a vertical hop. Table 6.2 shows the 12 performance requirements that the aforementioned objectives correspond to and their mapping to optimisation objectives.

The hopping height is defined to be the rise in the robot's CoM from the moment of lift-off to the apex of the hop. This definition provides us with a simple relationship between vertical lift-off velocity (parameter 'vcm1y' in Table 5.10) and hopping height, given by the formula

$$\text{height} = \frac{v_{\text{cm}}^2}{2\text{g}} = \frac{v_{\text{cm}}^2}{19.61}\,.$$

In all of the behavioural objectives the zero 'Angular Momentum' constraint is applied (see Section 5.3.2), and in the travelling hops (objectives 9–11) the desired horizontal CoM velocity is treated as an extra objective and not a constraint. Finally, the balancing ability measure is the angular velocity gain for the reasons explained in Section 5.5, which must have its magnitude maximised, and its value comes from a single calculation. The remaining objectives are all modelled as a squared error minimisation problem between the desired velocity and the outcome velocity of the CoM, at the moment of lift off.

In objective 8, Skippy is supposed to land and stop, meaning that there is not supposed to be a lift-off. However, the simulation code expects a lift-off, and uses the moment of lift-off as the trigger to stop the simulation. So a very slow lift-off velocity of 0.5 m/s is specified, which corresponds to a hopping height of 1 cm. Another special feature of objective 8 is that Skippy is making a transition from hopping to balancing behaviour, which requires that the CoM should be directly above the contact between the foot and the ground, so this is included as a second objective.

Performance objectives 8 to 12 all have two expressions listed in the objectives column in Table 6.2. They are implemented as multi-objective problems, but the final score is a single number computed from the values of the two expressions. Logically, this makes the optimisation process a single-objective problem; but it was discovered that the optimisation algorithm found good behaviours more quickly if it was given a multi-objective problem to solve.

Performance Envelope

In a series of single-objective preliminary experiments with Matlab's optimisation toolbox [68], designs that could get close to a 4 m (9 m/s) hop were also discovered, but they were so specialised for high hops that they failed to perform some of the other behaviours we required. So we settled for a 3.2 m target height, after a fall of 1.8 m as a reasonable compromise between hopping height

	Performance Objective	**Optimisation Objective(s)**
1	Hop from 0 (rest) to 2 m/s	min $(\text{vcm1y} - 2)^2$
2	Hop from -2 to 4 m/s	min $(\text{vcm1y} - 4)^2$
3	Hop from -4 to 6 m/s	min $(\text{vcm1y} - 6)^2$
4	Hop from -6 to 8 m/s	min $(\text{vcm1y} - 8)^2$
5	Hop from -8 to 6 m/s	min $(\text{vcm1y} - 6)^2$
6	Hop from -6 to 4 m/s	min $(\text{vcm1y} - 4)^2$
7	Hop from -4 to 2 m/s	min $(\text{vcm1y} - 2)^2$
8	Hop from -2 to 0.5 m/s	min $(\text{vcm1y} - 0.5)^2$ min $(\text{CoMx} - \text{ground contact x})^2$
9	Travelling hop from [0, -4] to [2, 4] m/s	min $(\text{vcm1x} - 2)^2$ min $(\text{vcm1y} - 4)^2$
10	Travelling hop from [2, -4] to [2, 4] m/s	min $(\text{vcm1x} - 2)^2$ min $(\text{vcm1y} - 4)^2$
11	Travelling hop from [2, -4] to [0, 4] m/s	min $(\text{vcm1x})^2$ min $(\text{vcm1y} - 4)^2$
12	Maximise balancing ability of the machine	max |Angular Velocity Gain|

Table 6.2: Mapping from performance objectives to optimisation objectives.

and versatility. Example behaviours demonstrating that the robot is close to its physical limits are presented in the results of Section 6.6.9.

Reachability

Skippy cannot reach 3.2 m (8 m/s) directly from a standing start. To achieve a 3.2 m hop Skippy must first reach a 1.8 m (6 m/s) height. This can be achieved by a series of successively increasing height hops. The values of these intermediate hops were also determined by single-objective experiments using Matlab. Exploratory optimisations revealed that the mechanism needed a minimum of three preliminary hops, and that suitable heights for these hops would be 0.4 m (2 m/s), 0.8 m (4 m/s) and 1.8 m. All three are close to Skippy's performance envelope as observed from the exploratory optimisations; few designs could make an initial hop significantly higher than 0.4 m; few could reach higher than 0.8 m after landing from a 0.4 m hop; and so on.

Similarly to the vertical hops, the travelling hops should also be a result of reachable states that the robot should come from and go to. Specifically, the robot starts a travelling hop of [2,4] m/s after a fall from a vertical hop of 0.8 m then achieves a limit cycle for that travelling velocity, and can go back to a vertical hop of 0.8 m again so it can eventually continue its operation.

Versatility

Besides the physically maximum vertical hop, we want the robot to display a plethora of other behaviours. For this reason, single-objective experiments were performed to determine a feasible travelling velocity for a Skippy mechanism, which is not close to its performance envelope in this category. In addition to that, because the robot has only one leg, it needs to be able to balance.

The balancing behaviour is not examined in these experiments, but the velocity gain is used (see Section 5.5), which has been shown as an effective measure of balancing ability.

Optimality of Sequence

For the successive hops up to 3.2 m (8 m/s) and back to a resting configuration we have selected a step of ± 2 m/s. This is not necessarily the optimal sequence to achieve the maximum hop or stop hopping after that. In this study we only seek any way to reach the performance envelope so we settled with these values when we discovered that they are feasible.

6.5.2 Selecting Parameters to Optimise

For the Mechanism Layer the eight design parameters presented in Table 6.3 were chosen to be optimised. The sensitivity analysis presented in Section 6.6.2 has shown that these parameters have a significant impact on the robot's performance for three selected performance objectives, which are: (1) the highest vertical hop, (2) the initial hop (starting from rest), and (3) the robot's balancing ability (AVG).

These parameters are: one kinematic parameter, one inertia parameter and six spring parameters. They were selected based on the following three criteria: (1) the sensitivity analysis results, (2) manufacturing/delivering lead time (i.e., some parts can be easily reordered or modified unlike other parts that may require a lot of time to be manufactured, such as the ring screw rod 5.3.1, or be delivered due to bureaucratic processes), and (3) a series of initial exploratory experiments where we experimented with various parameters in order to obtain a sufficient understanding of the system. In these experiments a larger number of parameters were allowed to vary, and we were able to determine which parameters have the largest influence on the system. A more elaborate discussion on the design parameters with the highest effect is presented in the sensitivity analysis section 6.6.2.

Mechanical Tolerances

The MOGA-II algorithm can handle both continuous and discrete variables; however, there are limits to precision to which mechanical components can be built. Ignoring these limits and allowing the optimiser to seek values beyond them can lead to over-optimisation of the theoretical model and amplifications of modelling errors. For example, with the batch of real fibreglass springs we obtained, tested and built our model on we observed variations between spring lengths of about 5 mm. This means that optimising for values less than this limit would result in a waste of resources, because the springs cannot be manufactured with such high tolerance. For this reason the 'step' value was set, so the search-space is reduced and the model does not get over-optimised.

In the Behaviour Layer, the initial foot angle ('q4i'), the initial nut position ('q10i'), which determines the hip angle, and all the 'Input Parameters' are optimised (Table 5.10). Overall, 15 behavioural parameters are optimised in the second layer, and eight design parameters in the first layer.

Name	Description	Lower Bound	Upper Bound	Step
d2toe	length of foot, ankle to toe tip [m]	0.19	0.26	0.001
torso_cx	torso+ centre of mass x coordinate [m]	0.35	0.45	0.001
as_L	ankle spring rest length [m]	0.15	0.22	0.005
as_Phi	ankle spring arc angle [rad]	0.9	1.2	0.005
as_F25	force to compress ankle spring 25% [N]	800	1300	5
ms_L	main spring rest length [m]	0.15	0.22	0.005
ms_Phi	main spring arc angle [rad]	1.2	1.9	0.005
ms_F25	force to compress main spring 25% [N]	800	1400	5

Table 6.3: List of eight parameters to be optimised in Skippy's mechanism. Step is the discrete distance between successive values that the optimiser is allowed to take. The symbol 'torso+' means the torso plus the crossbar, treated as a single rigid body.

6.5.3 Design of Experiments

To maximise the amount of qualitative information in the initial population, an appropriate set of designs and behaviours must be selected for the Mechanism and Behaviour Layer respectively. The aim is to create a trade-off between exploration and exploitation of the search space that will allow the algorithm to find at least one feasible design as well as to discover a diverse set of Pareto-optimal designs.

This is achieved by creating an initial population with seeds that are known to work for some performance objectives (discovered in preliminary single-objective optimisation experiments), and by adding new seeds generated by DOE techniques for exploration. As mentioned in Section 6.5.1, an initial exploration has resulted in a rough estimate of the potential of Skippy's mechanism in a series of single-objective optimisation experiments. In these preliminary studies, 11 different mechanisms and their optimal behaviours were discovered, for achieving each of the performance objectives presented in Table 6.2. Each of the discovered behaviours was used as an initial seed for the corresponding optimisation experiment of the Behaviour Layer.

For generating new designs, the *Incremental Space Filler* (ISF) algorithm [27] is used. Given the input variables and their bounds (see Table 6.3), ISF incrementally generates new designs by maximising the minimum distance from the existing points, using the euclidean norm. In this way, the population provides a better coverage of the search space.

Parameter Name	1st Layer (Mechanism)	2nd Layer (Behaviour)
Initial population	ISF	ISF + User-defined
Population Size	10	10
Number of Generations	30	30

Table 6.4: DOE parameters for the mechanism and the behaviour layers. ISF DOE technique is used to promote exploration of the search space by adding points that maximise the minimum distance between the existing points. 'User defined' means mechanisms or behaviours that were found from previous experiments and are known to perform well.

Mechanism Layer

In the absence of a single mechanism capable of achieving all of the 11 behavioural performance objectives, 10 initial designs were generated using the ISF Technique. These designs evolved for 30 generations, to produce 300 different Skippy mechanisms.

Behaviour Layer

Having obtained 11 behaviours and designs that are capable of achieving one of the performance objectives each, they were used as seeds in their corresponding behavioural optimisation of the Behaviour Layer. Each of these 11 optimisation experiments had an initial population of five seeds, out of which five were behaviours of mechanisms that can achieve only that objective, and the remaining five were generated using the ISF algorithm. Each behavioural experiment produced 30 generations, that resulted in 300 explored behaviours.

6.5.4 Global Optimisation

With the proposed approach 300 designs were discovered out of which 83 ($\sim$28%) were feasible and met all the performance objectives. A design is successful when the targeted vertical CoM velocity is within a range of ± 0.3 m/s from the desired value in both vertical and travelling hops, plus a tolerance of ± 0.3 m/s in the horizontal CoM velocity of the latter, in addition to not violating any of the constraints.

To facilitate the design selection process in the 12-dimensional space, a performance evaluation score is defined described by equations:

$$\begin{aligned} e_v &= \textstyle\sum_{i=1}^{8} |\dot{y}_i - \dot{y}_{\mathrm{T}i}| \\ e_t &= \textstyle\sum_{i=9}^{11} |\dot{x}_i - \dot{x}_{\mathrm{T}i}| + |\dot{y}_i - \dot{y}_{\mathrm{T}i}| \,. \end{aligned} \tag{6.1}$$

Value e_v is the total velocity error for vertical hops (objectives 1–8 in Table 6.2) and e_t is the total error travelling hops (objectives 9–11 in Table 6.2). $\dot{y}_i$ is the vertical lift-off velocity of the CoM for the best behaviour of performance objective i, and $\dot{y}_{\mathrm{T}i}$ is the target vertical lift-off velocity for the same performance objective. Similarly for $\dot{x}_i$ and $\dot{x}_{\mathrm{T}i}$, which are the final and target horizontal lift-off velocity of the CoM respectively. The total evaluation score for each design is defined as:

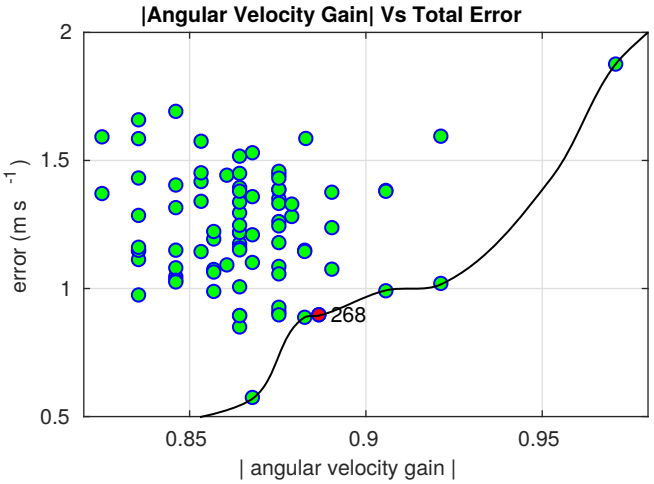

Figure 6.1: Plot of the magnitude of the AVG versus the total error as defined in Eq. 6.2. The design with the best trade-off between hopping ability and balancing was selected to be Design 268 (in magenta), which has a combination of high AVG value and low total error for the behavioural performance objectives. The Pareto front consists of the designs that the black line passes through.

$$e_{tot} = e_v + e_t \,. \tag{6.2}$$

Fit designs have a high magnitude of the AVG and low total error. Figure 6.1 shows the total error versus the magnitude of the AVG for all 83 successful designs discovered by the design optimisation approach. Designs close the bottom right of the plot are fitter than designs further away. Design 268 which is labelled and coloured in magenta has a high AVG value of 0.887 and can meet the specified performance requirements with a relatively low error.

The design located at the top right of the Figure 6.1, which has the highest AVG value, has also a very poor performance in hopping, which demonstrates the conflicting nature between balancing and hopping. Design 268 is used as an initial seed for the more advanced optimisation experiment of the next section. There, comparisons are made between Design 268 (the seed) and new optimal designs based on new and more advanced evaluation criteria. In addition, the optimal behaviours of the new optimal design are presented and discussed. The results presented in this section are published in Gkikakis and Featherstone [38].

6.6 Hopping, Travelling, Somersaulting and Balancing

In this section, the final result of the previous design study, which is a single design is used as a seed in a new and more advanced optimisation study. In this study the complete optimisation methodology proposed in Section 4.5 is presented together with a deeper discussion on the results and the evaluation criteria. Furthermore, several additions and changes were performed in the model of the mechanism and the performance objectives.

The changes in the model are summarised below.

1. A new model of the fibreglass spring is used with a more realistic hysteresis model. The immediate result of this is that Design 268 cannot meet any longer the 3.2 m hop due to the dissipated energy in the new spring model. The new maximum hopping height is now 3 m.

2. Losing energy is easier than gaining it, something that we also observed from the previous study. The number of successive hops required to come from the maximum hop to a halt has been reduced to three hops.

3. In exploratory experiments with design 268 and the new spring model, we discovered the capabilities of the new mechanism, and the performance objective values for the intermediate hops and travelling hops have changed.

4. Behaviours with angular momentum are explored by adding a new performance objective of a 2 m triple somersault after a 2 m vertical hop.

5. This experiment also explores stable limit cycles for vertical hops, by introducing a repeated vertical hop of 2 m.

The final result of this design optimisation study is a design and its optimal behaviours for achieving a set of desired performance objectives, and the optimal behaviours of the selected design are presented and analysed.

6.6.1 Performance Objectives

Table 6.6.1 shows the new performance objectives and their mapping to optimisation objectives. The new performance requirements are a result of the new spring model and knowledge gained from the previous experiments. To reach the new maximum hop of 3 m a minimum of three preliminary hops is needed, and the heights for these hops are 0.6 m, 1 m and 2 m.

Having reached a 3 m hop, Skippy needs to come back down again and stop. As losing energy is easier than gaining it, the new exploratory optimisations revealed that Skippy needs to bounce only twice before coming to rest, and that suitable heights for these two bounces would be 2 m and 0.6 m.

It was also discovered that Skippy can travel faster than the previous objective, despite the additional energy losses from the new spring model. This shows that the mechanism of Skippy has more potential in this objective, which could be explored in a future study where the performance envelope of travelling hops is sought. Specifically, the requirements for the travelling hop were increased from 0.8 m height and 1.6 m distance to a 1 m height and 2.3 m long hop.

In addition, a repeated vertical hop of 2 m is introduced. This performance requirement introduces a vertical hopping limit cycle, in addition to the travelling limit cycle. Furthermore, a 2 m triple somersault is added. Preliminary experiments showed that a single or double somersault can be achieved easily by various mechanisms, but a quadruple somersault was very difficult to achieve, so

	Performance Objective	**Optimisation Objective(s)**
1	Hop from 0 (rest) to 3.4 m/s	$\min (\text{vcm1y} - 3.4)^2$
2	Hop from -3.4 to 4.5 m/s	$\min (\text{vcm1y} - 4.5)^2$
3	Hop from -4.5 to 6.3 m/s	$\min (\text{vcm1y} - 6.3)^2$
4	Hop from -6.3 to 7.7 m/s	$\min (\text{vcm1y} - 7.7)^2$
5	Hop from -7.7 to 6.3 m/s	$\min (\text{vcm1y} - 6.3)^2$
6	Hop from -6.3 to 6.3 m/s	$\min (\text{vcm1y} - 6.3)^2$
7	Hop from -6.3 to 3.4 m/s	$\min (\text{vcm1y} - 3.4)^2$
8	Hop from -3.4 to 0.5 m/s	$\min (\text{vcm1y} - 0.5)^2$
9	Travelling hop from [0, -4.5] to [2.5, 4.5] m/s	$\min (\text{vcm1x} - 2.5)^2$ $\min (\text{vcm1y} - 4.5)^2$
10	Travelling hop from [2.5, -4.5] to [2.5, 4.5] m/s	$\min (\text{vcm1x} - 2.5)^2$ $\min (\text{vcm1y} - 4.5)^2$
11	Travelling hop from [2.5, -4.5] to [0, 4.5] m/s	$\min (\text{vcm1x})^2$ $\min (\text{vcm1y} - 4.5)^2$
12	Triple somersault from [0, -6.3] to [0, 6.3] m/s	$\min (\text{vcm1y} - 6.3)^2$ $\min (\text{hcm1} - 3.5)^2$
13	Maximise balancing ability of the machine	$\max \lvert\text{Angular Velocity Gain}\rvert$

Table 6.5: The performance objectives and their mapping to optimisation objectives.

we settled for 2 m triple back-flip. This performance objective introduces behaviours with high angular momentum, something that was not examined in the previous study. To achieve the somersault, an angular momentum that is sufficient for the robot to perform three full self rotations about the sagittal plane is defined. This value assumes that the robot is in a neutral configuration (hip angle is in the midpoint of its fully closed and open configuration) during the flight phase, which can be easily achieved by a flight-phase controller. The time that the robot spends in the air is defined as:

$$t = \frac{2\,v_{cm}}{g} = \frac{2\,v_{cm}}{9.81}\,, \tag{6.3}$$

and for an initial velocity of 6.3 m/s to achieve a 2 m vertical hop, this is about 1.28 s.

This study follows the same philosophy as the previous one. Specifically, versatility is improved by adding new behaviours, which are the somersault and a repeated vertical hopping cycle; minimality (the number of steps to come to a halt from the maximum hopping height has been reduced to three steps instead of four); and reachability except for the case of stopping from the 2 m somersault.

The travelling hops and the vertical hops are mapped to optimisation objectives in the same way that was presented in the previous section, and the somersault is treated as a bi-objective optimisation problem by minimising the errors between the desired and outcome angular momentum, and desired and outcome lift-off vertical CoM velocity.

6.6.2 Sensitivity Analysis

In this section, a sensitivity analysis is performed according to the methodology presented in Section 4.5. This analysis serves the following purpose: 1) to gain a deeper understanding of the examined

system, 2) to identify the design parameters with the highest influence in Skippy, and 3) to reduce the dimensionality of the search space by removing from the experiment parameters with low influence, which will lead to less computations.

DOE

To perform the analysis, a set of designs is generated so that the input design variables (or factors in the context of sensitivity analysis) have low correlation between them. Then, sensitivity analysis is used to identify interaction effects of factors on responses (outputs).

To obtain an appropriate initial population for the experiment, a 2-level Reduced Factorial algorithm is used to generate 1024 designs. Due to the high dimensionality of the search space and the high cost of simulations the Full Factorial algorithm (see Section 4.5.2) is prohibitively expensive.

Correlations between input values can lead to biased results that are unreliable. To validate the efficiency of the sampling algorithm the co-linearity indices of the sampled data are calculated. Index values close to one indicate a well-distributed dataset over the search-space [42]. Insufficient sampling of the search space or correlated input variables could yield co-linearity indices with values larger than one. The mean value of the co-linearity indices of the generated dataset is ~ 1.001.

Algorithm

For estimating the interaction effects the *Smoothing Spline Analysis of variance* [42] (SS-ANOVA) approach is used. SS-ANOVA is a non-parametric statistical approach based on the classical ANOVA, and is used for identifying main and interaction linear effects. These methods are used to quantify the interaction effect with a single value, which represents the effect of each input variable to the global variance (the objective functions in the examined case) as a percentage. Details about SS-ANOVA can be found in Gu [42].

Selecting Parameters

A sensitivity analysis is performed for 13 design parameters, and for three performance objectives. The design parameters have all been previously introduced in Section 5.3.1, and are summarised in Table 6.6. Preliminary optimisation experiments have indicated that the selected parameters have potentially a significant influence on the robot's performance.

As shown in Figure 5.7, the curvature of the fibreglass leaf springs (parameters 'as_Phi' and 'ms_Phi') affects their regression behaviour, which means greater stored elastic energy for a given stroke and maximum force (which affects hopping), and greater stiffness at low force levels, which affects balancing. Parameters 'as_F25' and 'ms_F25' define the maximum force of the spring at the given stroke (at 25% compression from its rest length) and therefore the amount of stored elastic energy, which is crucial for performing hops.

Parameter 'torso_cx' specifies the horizontal location of the torso's centre of mass (CoM) which holds a significant part of the robot's total mass (2.2 kg out of 3 kg). This parameter affects the ground

Name	Description	Lower Bound	Upper Bound
Kinematic Parameters and Limits			
rtoe	radius of toe [rad]	0.25	0.35
d2toe	length of foot (body 4) [m]	0.20	0.26
d05	lever length [m]	0.1	0.11
torso_cx	torso+ centre of mass x coordinate [m]	0.35	0.45
Ankle spring parameters			
as_L	ankle spring rest length [m]	0.14	0.26
as_Phi	ankle spring arc angle [rad]	0.5	2
as_F25	force to compress ankle spring 25% [N]	950	1250
aslv_len	ankle spring lever length [m]	0.04	0.05
Main spring parameters			
ms_L	main spring rest length [m]	0.14	0.26
ms_Phi	main spring arc angle [rad]	0.5	2
ms_F25	force to compress main spring 25% [N]	950	1250
Ankle end stop			
q5es	ankle end stop engagement angle [rad]	0.75	0.95
Hip end stop			
q67es	hip end stop engagement position [rad]	0.1	0.2

Table 6.6: Design parameters that were taken into consideration in the sensitivity analysis. These parameters have shown to affect the robot's performance in preliminary optimisation studies, and hence have been chosen to be examined in a thorough analysis.

reaction forces and as a result the final outcome of the stance phase. Parameter 'd2toe' specifies the length of the foot measured from the toe to the ankle, and affects the direction of the ground reaction force making it also an important parameter for hopping. Parameters 'd2toe' and 'torso_cx' both affect the magnitude of the angular velocity gain (see Section 5.5). The effect of foot curvature (parameter 'rtoe') on legged locomotion has been the topic of interest in other disciplines, such as in biomechanics of human walking in the work of Sichting et al. [100], where it has been shown that this parameter can result in lower work requirements during walking, and hence facilitate legged locomotion.

The end stops are used for preventing the robot from reaching undesired configurations by defining the range of motion of the joints, and for dissipating energy upon activation, hence their activation positions play a crucial role in the performance of the mechanism. In preliminary experiments different Skippy mechanisms engage the hip end stop during behaviours that result in the 3 m hop. Early activation of the end stops may limit the performance of the mechanism; and late activation may lead to over-extension of the robot's joints.

Other parameters that were investigated in preliminary experiments but not in the sensitivity analysis study include the hip and ankle end stop parameters, and the ankle rest angle. The leg length (parameter 'p2x' in Table 5.2) was always kept fixed in order to fix the overall scale of the robot; otherwise the optimiser would try to make the whole robot bigger.

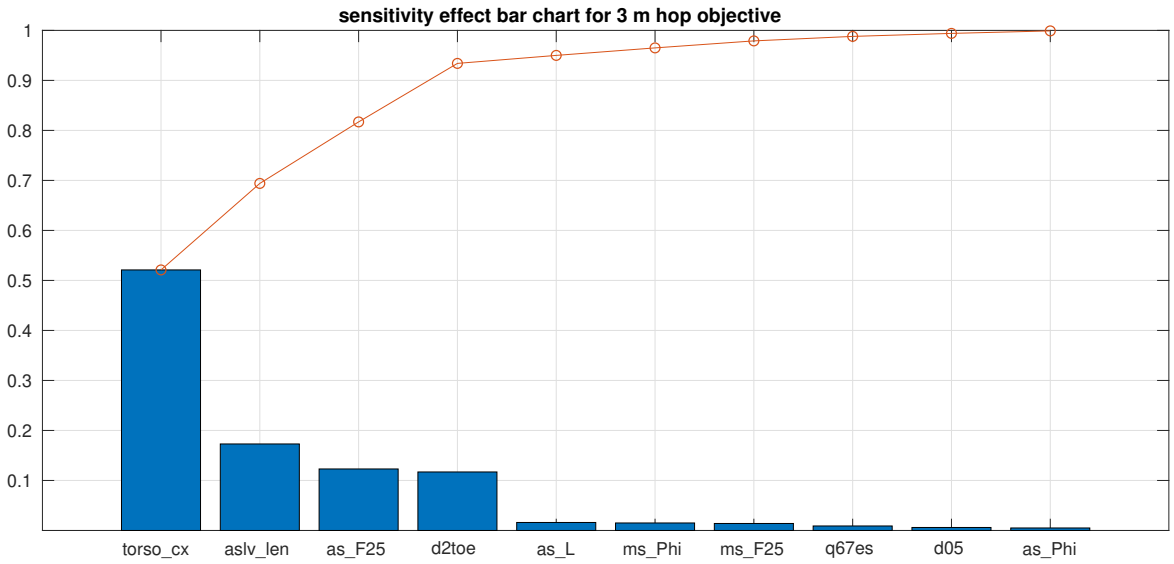

Figure 6.2: Sensitivity-effect bar chart for a hopping height of 3 m from a 2 m fall.

Results

The analysis was performed for three objectives, which are: (1) the AVG, (2) the initial hop (from rest to 0.6 m vertical hop), and (3) the high hop (from -2 m to a 3 m vertical hop). These are corresponding objectives 1, 4 and 13 in Table 6.5. The high hop is an important objective because it is the behaviour where the performance envelope of the robot is sought, and potentially the most demanding. The initial hop is examined to make a comparison between the effects of the design parameters in both high and low hops. Finally, the angular velocity gain describes the balancing ability of the robot, which is also an important behaviour, and is the product of a simple computation (see Section 5.5).

Figures 6.2–6.4 present the main results of the analysis, which are the sensitivity-effect bar charts. In these charts, each bar represents the percentage of interaction that a corresponding parameter has on the variance of the examined performance objective. The sum of all interaction values is equal to one. For each chart, only the most influential parameters are presented, which are the parameters that their cumulative effect sums up to 99%.

Figure 6.2 presents the effect of 10 parameters in the behaviour to achieve a vertical hopping height of 3 m after a 2 m hop. The parameters with the highest effect are: (1) the torso's centre of mass x coordinate ('torso_cx'), (2) the lever distance of the ankle spring to the leg ('aslv_len'), (3) the force at 25% of the ankle spring (as_F25), and (4) the length of the foot (d2toe). Parameter 'torso_cx' has the highest influence ($\sim$ 52%) in the maximum hop. Finally, one can also observe that the activation point of the hip end-stop ('q67es') has a small effect on the performance of the robot because it is engaged in some behaviours.

Figure 6.3 shows the effect of the eight most influential parameters in the behaviour to achieve an initial hopping height of 0.6 m. The challenging aspect of this behaviour is to build the required momentum to lift off from the ground, when starting from rest. Once the robot achieves its first

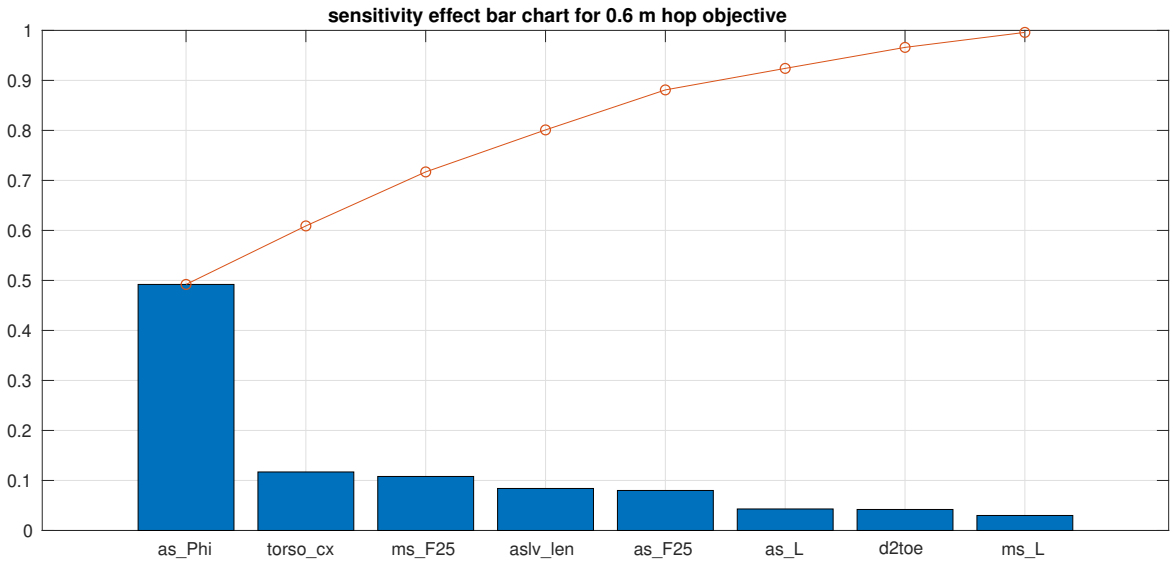

Figure 6.3: Sensitivity-effect bar chart for achieving a hopping height of 0.6 m from rest.

hop, the system can then recycle stored energy from previous hops to increase its hopping height. According to these results, the parameter with the highest effect (accounts for almost 50% of the overall effect) in the initial hop is the curvature of the ankle spring ('as_Phi'). This parameter, affects the regression behaviour (the degree to which the stiffness decreases with increasing compression) of the ankle spring. A highly-regressive spring has high initial stiffness, and hence is more challenging to compress, especially from a state of low momentum, which is the case in the initial hop.

Figure 6.4 presents the effect of three design parameters in the magnitude of the AVG (the other parameters do not affect its value). One can observe that the x coordinate of the torso's centre of mass has the highest impact (86%) making it the most important parameter in Skippy for achieving a high-balancing skill. The 4-bar lever length ('d05') has an effect of $\sim 11\%$, followed by the foot length which contributes at about 4% in the variance of the magnitude of the AVG.

Conclusion

To conclude, the sensitivity analysis results indicate that parameter 'torso_cx' has a high impact in the three examined performance objectives. Furthermore, the spring parameters have a high contribution to the performance of the high and initial hops. In addition, the length of the foot ('d2toe') also has a significant influence in both examined behaviours, and the lever length ('d05') appears to have a significant impact in the balancing behaviour of the robot (AVG) and a relatively small impact in the 3 m hop. These findings are taken into consideration for selecting the design parameters to be optimised during the following optimisation experiments.

In order to facilitate the CAD design of the robot, we want to make it as modular so possible, so we can easily perform changes without having to make major design modifications. For this reason, we decided to keep fixed some of the examined parameters. Parameters 'as_L' and 'ms_L' were kept

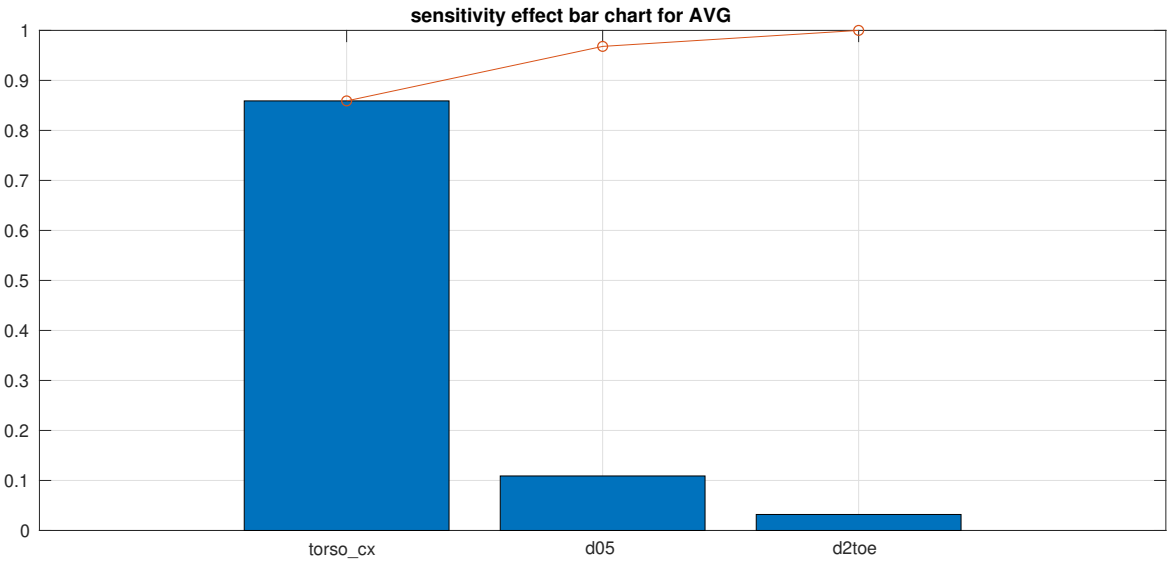

Figure 6.4: Sensitivity-effect bar chart for the AVG.

fixed at 0.195 m, and only the curvature and specified force for both springs are optimised. Parameter 'aslv_len', which is the attachment point of the ankle spring to the ankle is also kept fixed, because it affects the design of the ankle. For the same reason the ankle and hip end stop engagement positions are now constants. By varying the spring arc angle and stiffness, we are able to modify the robot simply by replacing them with new ones, without the need of extra modifications.

6.6.3 Global Optimisation

Having obtained sufficient information for setting up the design optimisation experiment, a global optimisation is performed for obtaining rough solutions. The objectives of this optimisation study are presented in Section 6.5, the constraints are the same as the ones defined in Section 6.2, and the experiment is performed with the MOGA-II algorithm (see Algorithm 1).

Selecting Parameters to Optimise

The parameters chosen to be optimised in the global optimisation study are presented in Table 6.7. In this experiment, two kinematic parameters, one inertia parameter and four spring parameters are optimised, while the remaining design values are kept fixed. The importance of these parameters in the design's performance has been demonstrated in the sensitivity analysis study (see Section 6.6.2). Parameter 'd05' is now optimised, which is the lever length between the 4-bar linkage and the main actuator (see Figure 5.4). This part effectively controls how much torque can be generated at the hip.

Name	Description	Lower Bound	Upper Bound	Step
d2toe	length of foot, ankle to toe tip [m]	0.22	0.26	0.001
d05	4-bar lever length [m]	0.1	0.11	0.0005
torso_cx	torso+ centre of mass x coordinate [m]	0.36	0.44	0.001
as_Phi	ankle spring arc angle [rad]	0.9	1.2	0.01
as_F25	force to compress ankle spring 25% [N]	950	1150	2
ms_Phi	main spring arc angle [rad]	1.6	1.9	0.01
ms_F25	force to compress main spring 25% [N]	950	1250	2

Table 6.7: List of seven parameters to be optimised in Skippy's mechanism. Step is the discrete distance between successive values that the optimiser is allowed to take. The symbol 'torso+' means the torso plus the crossbar, treated as a single rigid body.

Parameter Name	1st Layer (Mechanism)	2nd Layer (Behaviour)
Initial population	SOBOL	SOBOL + User-defined
Population Size	10	10
Number of Generations	40	100

Table 6.8: DOE parameters for the mechanism and the behaviour layers. SOBOL [27] DOE technique is used to promote exploration of the search space by generating points with maximal distance between them. 'User defined' means behaviours that were found from previous experiments and are known to meet the objectives.

Design of Experiments

In this experiment a different DOE algorithm is selected to generate the initial population, which consists of Design 268 (will now be called Design 0) and nine new designs. Specifically, the quasi-random algorithm called *SOBOL* [27] is used. This algorithms seeks to fill the search-space in a uniform way by generating samples that are maximally distanced from each other. In this way, the clustering of solutions that happens with a random algorithm is avoided, and a more uniform sampling of the search space is achieved. The DOE parameters are summarised in Table 6.8.

Mechanism Layer—the new seed (design 0) is used as a part of the initial population for exploitation of the current available information, and an additional nine designs are generated via SOBOL for exploring the design space.

Behaviour Layer—the optimal behaviour of design 0 for each performance objective is used as a seed for the corresponding optimisation experiment of the second layer. In addition to that, nine more behaviours are generated using the SOBOL technique.

6.6.4 Evaluation criteria

To facilitate the comparison between designs, the performance objectives are grouped into four categories as follows:

1. vertical hops (objectives 1–8),
2. travelling hops (objectives 9–11),
3. somersault (objective 12), and
4. balancing (objective 13).

For each of the first three categories an error measure and an energy measure are defined as follows:

$$\begin{aligned} e_\mathrm{v} &= \textstyle\sum_{i=1}^{8} |\dot{y}_i - \dot{y}_{\mathrm{T}i}| & E_\mathrm{v} &= \textstyle\sum_{i=1}^{8} E_i \\ e_\mathrm{t} &= \textstyle\sum_{i=9}^{11} |\dot{x}_i - \dot{x}_{\mathrm{T}i}| + |\dot{y}_i - \dot{y}_{\mathrm{T}i}| & E_\mathrm{t} &= \textstyle\sum_{i=9}^{11} E_i \\ e_\mathrm{s} &= |\dot{y}_{12} - 6.3| + C|h_{12} - 3.5| & E_\mathrm{s} &= E_{12}\,. \end{aligned} \tag{6.4}$$

In these equations, e_v, e_t and e_s are the error measures for the first three categories; E_v, E_t and E_s are the energy measures; $\dot{x}_i$ and $\dot{y}_i$ are the CoM velocities at lift-off (from parameters 'vcm1x' and 'vcm1y') of the best behaviour for objective i; $\dot{x}_{\mathrm{T}i}$ and $\dot{y}_{\mathrm{T}i}$ are the target values for $\dot{x}_i$ and $\dot{y}_i$; h_{12} is the robot's angular momentum at lift-off (from 'hcm1') of the best behaviour for objective 12; E_i is the energy drawn from the battery during the execution of the best behaviour for objective i; and C is a conversion factor to allow the addition of an angular momentum to a linear velocity. A suitable value for C is $1/(m_\mathrm{tot} r_\mathrm{fl})$, where m_tot is the total mass of the robot and r_fl is its radius of gyration in a suitable flight-phase configuration. There is a range of possible values for r_fl, depending on the hip angle, including one that makes C numerically equal to 1. So we chose the value $C = 1\,\mathrm{kg}^{-1}\mathrm{m}^{-1}$.

In addition to the above, an overall error and energy measure is defined as follows:

$$e_\mathrm{o} = e_\mathrm{v} + e_\mathrm{t} + e_\mathrm{s}, \qquad E_\mathrm{o} = E_\mathrm{v} + E_\mathrm{t} + E_\mathrm{s}\,. \tag{6.5}$$

And finally, the measure for balancing is the magnitude of the robot's angular velocity gain, as it was for the previous experiment. For this one measure, bigger is better. For all of the others, smaller is better.

The total energy consumption is used as an extra evaluation criterion during the final selection, but was not taken into consideration by the optimiser during the optimisation experiment.

6.6.5 Performance Comparisons

The optimisation framework evaluated 400 different Skippy mechanisms, out of which 34 (8.5%) of them met the desired performance requirements. A design was considered successful if none of the constraints were violated, and the optimisation objectives were within a tolerance of ± 0.2 m/s or

	Initial Seed Des. 0	Best Performance Des. 499	Hopping Des. 1784	Somer. Des. 1011	Travelling Des. 2363	Balancing Des. 2371
E_o	1005 J	+1%	−6.4%	−5%	−10.9%	−1.7%
E_v	582 J	+1.6%	−3.4%	−3.7%	−12%	−1%
E_s	190 J	+0.3%	−12.3%	−7.7%	−7.2%	−1.1%
E_t	233 J	0%	−9%	−6.3%	−11.2 %	−5.7%
AVG	0.894	0%	+1.9%	+0.6 %	−8.1%	+3.2%
e_o	0.561 m/s	−23%	+276%	+171%	+316%	+243%

Table 6.9: Performance comparison between the initial optimisation seed (design 0) and five selected designs discovered by the proposed framework. Each of the five discovered design scores are presented as a percentage relative to Design 0 scores.

$\pm 0.2\,\mathrm{kgm^2s^{-1}}$ from the desired lift-off conditions. These are stricter tolerances than the ones in the previous experiment.

The framework discovered designs that are significantly better than Design 0 in all categories. These results demonstrate the capabilities of the proposed optimisation approach to efficiently explore the trade-offs between several conflicting performance objectives. From the successful designs, six are selected to be compared. These designs are: (1) the seed (0), (2) the best vertical hopper (1784), (3) the best travelling hopper (2363), (4) the best somersaulter (1011), (5) the best balancer (2371), and (6) design (499) with the highest performance (lowest error in performance objectives).

Figure 6.5 presents 6 scatter plots with all the successful designs. This figure shows plots with all the performance criteria of interest, plotted against each other. The selected designs are coloured in magenta, and the Pareto front in each plot consists of the designs that the black line passes through. Table 6.9 shows the energy spent by the battery, AVG and error measure of the seed, as well as the percentage variations from the seed of the five selected designs. For example, design 2363 consumes 10.9% less energy than the seed in its overall performance (E_o), but it does not perform well at balancing, and is also under-performing in low hops. The best design at somersaulting (design 1011) is 5% more energy efficient, better at balancing than design 0, and can achieve all objectives with a relatively small error; however, it can only reach a hopping height of 2.85 m (7.48 m/s) after a landing of 2 m (−6.3 m/s).

Design 499 achieves the performance objectives with a 23% percent less total error than the seed, by having only a 1% variation in one of the design parameters, which is the main spring's arc angle. This has consequences for the design of the real robot. In experiments performed with a real batch of fibreglass springs, it was discovered that a variation of 3% in stiffness exists between the stiffest and the least stiff spring. Furthermore, one possible justification of the fact that design 499 requires more energy to achieve its objectives could be the fact that it achieves all of the desired performance objectives with a very small error, while other designs did not come as close, such as the best somersaulter in the high hop case. These results indicate that further investigation is required, and

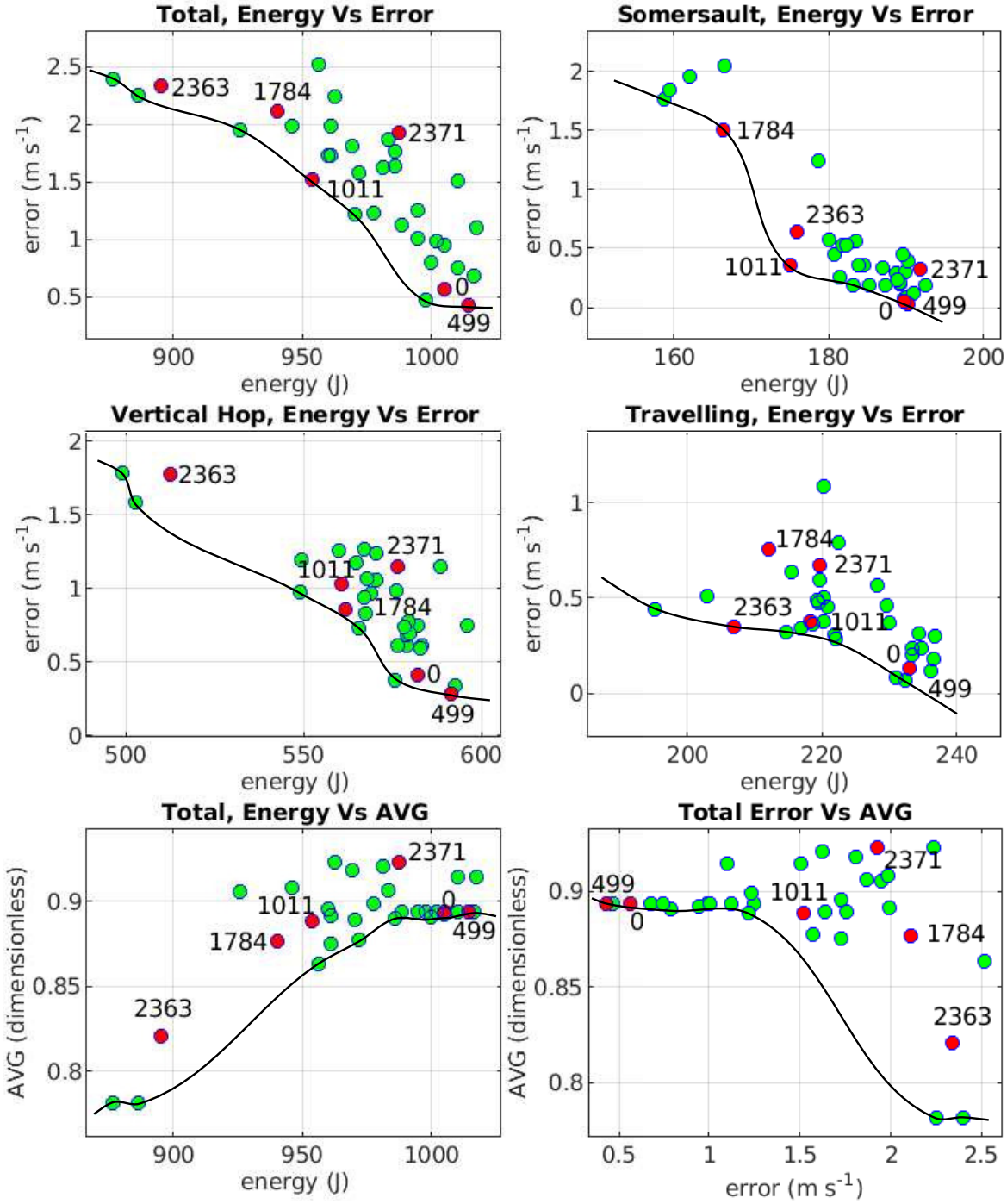

Figure 6.5: Plotting the error measure against energy consumption for each of the categories defined in Section 6.6.4, for all the feasible designs. The selected optimal designs for each category, plus the initial seed are coloured in magenta.

	Des. 0	Des. 499	Des. 1784	Des. 1011	Des. 2363	Des. 2371	Bounds
d2toe [m]	0.24	0.24	0.236	0.247	0.223	0.225	[0.22, 0.26]
d05 [m]	0.105	0.105	0.1055	0.105	0.1085	0.102	[0.1, 0.11]
torso_cx [m]	0.4	0.4	0.393	0.4	0.374	0.397	[0.36, 0.44]
as_Phi [rad]	1.02	1.02	1.2	1.02	0.95	0.92	[0.9, 1.2]
as_F25 [N]	1072	1072	1084	1106	1050	976	[950, 1150]
ms_Phi [rad]	1.9	1.88	1.82	1.89	1.74	1.69	[1.6, 1.9]
ms_F25 [N]	1184	1184	1188	1184	1074	1182	[950, 1250]

Table 6.10: Table with selected designs and their parameters with their respective bounds, as found by the optimisation framework. Descriptions of the parameters can be found in Tables 5.2 and 5.6.

the conclusion is that refinement optimisation and robustness analysis should be strongly considered before selecting which design to build.

Table 6.10 presents the values of the design parameters of the seed and selected designs. One can see that the main spring's force at 25% ('ms_F25') is almost exactly the right value for the performance requirements. The remaining six parameters covered a large portion of the bounded design space, which shows that the framework has adequately explored it. The ankle-spring and main-spring arc angles vary within ranges of 0.28 rad and 0.21 rad, respectively; the length of the foot varies within a 2.4 cm range; and the lever length of the 4-bar linkage varies within a 6.5 mm range. The ankle spring's force at 25% varies within a 130 N range; and the corresponding parameter for the main spring varies within a 108 N range. However, in only one design is this last parameter significantly different from Design 0, and that is the best travelling hopper (2363).

One reason that the framework did not discover designs with significantly better trade-offs than the seed could be that the latter was already optimal for a very similar set of performance objectives. In addition, besides the maximum hopping height, which was shown to be close to the mechanism's performance envelope, the rest of the performance requirements were decided based on the performance of the seed. A more complete study that investigates also the optimality of the hopping sequence could potentially yield a different outcome; however, this results in a higher-dimensional search space, and, as explained in the Discussion Chapter, the study was constrained by the very high computational cost and time for the models to execute, so time did not allow for this experiment, which could be part of a future work.

6.6.6 Refinement Optimisation

Having obtained the results of a rough optimisation, a refinement optimisation is performed to discover the true potential of a set of the discovered mechanisms that seem to be the most promising. Given these mechanisms, an extra optimisation round is performed to discover the optimal performances of the selected designs in each of the performance objectives—except the AVG, which remains constant because the design parameters do not vary in this experiment.

Parameter Name	1st Layer (Mechanism)	2nd Layer (Behaviour)
Initial population	User-defined	User-defined
Population Size	5	10
Number of Generations	0	200

Table 6.11: DOE parameters for the mechanism and the behaviour layers for the refinement optimisation experiment. In this experiment the optimal behaviours of each design are sought, and hence design parameters remain constant. 'User defined' means mechanisms that were found from previous experiments and are known to meet the objectives.

	Best Performance Des. 499	Hopping Des. 1784	Somer. Des. 1011	Travelling Des. 2363	Balancing Des. 2371
E_o [J]	1015	953	971	913	998
$E_o \pm [\%]$	0	$+1.3$	$+1.7$	$+1.9$	$+1$
e_o [m/s]	0.451	0.795	0.916	1.194	0.853
$e_o \pm [\%]$	0	-62	-40	-49	-55
AVG	0.894	0.91	0.899	0.821	0.922

Table 6.12: Performance comparison between the five selected designs after the refinement optimisation experiment. The percentage values show the difference between the refined and prior performance.

DOE

The DOE parameters are summarised in Table 6.11. This experiment is a behaviour-only optimisation study with the aim to identify the true potential of the selected designs. The set of examined designs consists of the five selected designs presented in the previous section, and are listed in Table 6.10.

The initial behaviours of each objective are the optimal behaviours discovered in the global optimisation step. For each design, and for each performance objective 10 behaviours are selected and are used as the initial population set. This is an exploitation experiment so no randomly generated designs are added.

Algorithm

In this step, two different algorithms were explored: 1) Matlab's Interior Point algorithm [68], and 2) modeFrontier's MOGA-II algorithm [27], which is explained in algorithm 1. The latter produced better results, which are presented in the following section. A possible explanation for the fact that a global optimisation algorithm performed better than a local optimisation algorithm in a refinement experiment could be due to the potentially high roughness of the examined objective functions, which results in multiple local optima.

Results and Discussion

Table 6.12 summarises the results of the refinement optimisation experiment. The overall errors of all the designs except design 499 were significantly reduced, which demonstrates the effectiveness of this step. Design 499 still has the highest performance but showed very little improvement (close to zero). This can be attributed to the fact that this design has a small difference when compared to design 0 (the initial seed), whose behaviours were already refined and were used as a starting point for the behaviour optimisation layer. More importantly, after this step all five designs can now achieve the desired performance objectives with a small error margin. For example, design 1011, which could achieve a 2.85 m (7.48 m/s) hop in the previous study, can now reach 2.99 m (7.656 m/s) after a 2 m (-6.3 m/s) fall.

An interesting observation is the apparent inverse proportional relationship between energy efficiency and performance. The results of Table 6.12 show that the overall error decreased and the total energy spent by the batteries increased. This result shows that more precise and restrictive behaviours require more energy to achieve, and hence more effort. One can also observe that the design with the largest error value (design 2363) also spends the least amount of energy. This apparent relationship between effort and performance objective accuracy requires further investigation; however, this result is one of the reasons that energy consumption was not included as a performance objective during the global optimisation study. If the above assumption is true, the optimiser would have created a Pareto front that optimises the trade-off between energy consumption and objective accuracy; something undesired because in this study high performance is preferred over energy efficiency.

Another explanation for this result could be that the required objectives are not the optimal ones for the given mechanism. For example, in the majority of the objectives the robot is required to have zero angular momentum at lift-off; perhaps the mechanism would perform significantly better if a small amount of non-zero angular momentum was considered acceptable.

With a refinement optimisation the maximum potential of a mechanism is obtained; however, how far away the result could be from the performance of the real robot, in the real world that is full of imperfections and uncertainties? To answer this question a robustness analysis must be performed.

6.6.7 Robustness Analysis

A robustness analysis is performed to take into account imperfections in the real world. This is an important step in the proposed design optimisation approach. As mentioned in Part I Section 4.5.6, with this study the designer can examine and evaluate designs based on their robustness in expected design errors (e.g., manufacturing errors) or inaccuracies in model parameters.

Methodology

The methodology described in Section 4.5.6 is applied for the robustness analysis and is summarised as follows.

Initial Designs	Variables	Robust Designs	Distribution	Sampling Algorithm
5 User-defined	15	50	Gaussian	LHS

Name	Description	Standard Deviation (σ_i)
	Kinematic Parameters and Limits	
rtoe	radius of toe [m]	0.0001
d2toe	length of foot, ankle to toe tip [m]	0.0005
d05	4-bar lever length [m]	0.0001
	Dynamic Parameters	
torso_cx	torso+ centre of mass x coordinate [m]	0.0003
	4-bar Parameters	
bar4_a	length of 4-bar segment (in torso) [m]	0.0001
bar4_b	length of 4-bar segment (in lever, body 6) [m]	0.0001
bar4_c	length of 4-bar segment (in leg) [m]	0.0001
bar4_d	length of 4-bar segment (follower, body 8) [m]	0.0001
	Ankle Spring Parameters	
as_Phi	ankle spring arc angle [rad]	0.002
as_F25	force to compress ankle spring 25% [N]	10
as_L	ankle spring rest length [m]	0.0005
	Main Spring Parameters	
ms_Phi	main spring arc angle [rad]	0.002
ms_F25	force to compress main spring 25% [N]	10
ms_L	main spring rest length [m]	0.0005
	Ring Screw	
effi	ring screw efficiency	0.003

Table 6.13: Top table: summary of hyper parameters in the robustness optimisation experiment. Bottom table: list of 15 parameters examined in the robustness analysis. Random samples are drawn from 15 Gaussian distributions with means equal to each examined design's parameter values, and a standard deviation that represents expected errors that may appear in the real design. For each examined design 50 random designs are generated using a Latin Hypercube Sampling algorithm (LHS).

1. Select designs to evaluate from an existing Pareto front.
2. Identify parameters with uncertainty in the real world.
3. Define stochastic design variables and sampling algorithm.
4. Define objectives and constraints.
5. Run the optimisation.

Select designs to evaluate—the five selected designs presented in the previous section are studied.

Identify parameters with uncertainty—in this step the parameters to be included in the robustness study are identified. These parameters are presented in Table 6.13. In addition to the seven parameters that were examined in the global optimisation experiment (see Section 6.6.3), eight new parameters are added for the robustness analysis. These parameters are expected to have random variations in the

real world, and hence these effects should be examined. The source of uncertainty for each of these parameters is explained below.

1. **Spring force**—given a batch of fibreglass springs, a series of tests was performed (see Section 5.3.1). In these experiments we discovered that springs of the same batch may have a variation up to 3% in stiffness between the stiffest and the least stiff spring.

2. **Spring arc angle and length**—these parameters have also been measured to have variations between springs of the same batch, which can be quantified by the values presented in Table 6.13.

3. **4-bar segments**—these parts are made of aluminium, which in general can be manufactured in high precision; however, the 4-bar carries large forces of up to 3000 N, and hence the effect of small variations in their lengths might have a significant impact. In addition, these parts (see Figures 5.2 and 5.4) are held together with ball bearings, which have a small amount of radial play.

4. **Foot length and radius of the toe**—the foot will also be manufactured from aluminium; however, the part of the foot that is in contact with the ground will be covered by a layer of high-friction rubber to prevent slipping, which may have a slight effect on the length and curvature of the foot.

5. **Torso CoM x coordinate**—this parameter is crucial for both hopping and balancing (see Section 6.6.2), and in the real robot it can only be estimated up to a certain precision; so possible deviations of this parameters from its theoretical value should be examined.

6. **Ring screw efficiency**—this parameter was experimentally estimated, so in reality the actual efficiency might be slightly different.

Define design distribution and sampling algorithm—to simulate and examine the effects of variations on the design's performance a random distribution is defined around each examined design parameter, and random samples are drawn from it. The sampling process produces random new designs with expected variations on the vicinity of the original design. To verify the robustness of a design several samples around an initial design are generated and evaluated.

A Gaussian distribution

$$X_i \sim \mathcal{N}(\mu_i, \sigma_i^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu_i)^2 \big/ 2\sigma_i^2} \tag{6.6}$$

is defined for each examined design parameter X_i in Table 6.13, with a mean equal to the value of that parameter for the examined design (see Table 6.10) and a standard deviation describing the expected uncertainty of this parameter in the real world. Table 6.13 presents the design parameters and their standard deviation. For each of the five designs 50 new designs are generated using a *Latin Hypercube Sampling* algorithm [94].

Define objectives and constraints—the designs are subject to the same constrains presented in Section 6.2. The objectives also remain the same for this evaluation-only experiment; however, instead of 13 objective values the output now is a set of 13 distributions (one for each performance objective Table 6.5) for each of the five designs.

The aim is to obtain designs with low standard deviations in their objective values, or in other words designs that are robust to random variations of the parameters presented in Table 6.13. It is up to the designer to determine what would be the threshold between a valid and invalid robust design.

DOE

Designs—for robustness analysis the five selected designs presented in Section 6.6.5 were selected.

Behaviours—the optimal behaviours for each performance objective were discovered in the refinement optimisation step, so they are used as an initial population in the robustness analysis study. The aim of this study is to exploit available information rather than explore, so no random designs are added in the initial population set. Each design has a different set of initial population behaviours with 10 behaviours for each performance objective.

Results

Figures 6.6 and 6.7 show 12 box-and-whisker plots, which allow the visualisation of the degree of dispersion (spread) and skewness in the data, and outliers. A plot is generated for each objective (except the AVG) and contains one box-whisker for each of the five designs. These plots provide a visual representation of the variations of the statistical samples (50 for each examined design). The horizontal red lines in each box exhibit the median of the samples; each box length is the interquartile range (IQR), which is the distance between the upper and lower quartiles (contains 50% of the observations); the whisker's length is determined by the variance of the data and is defined by the values of the largest and lowest samples observed at 1.5 times the IQR; and red crosses represent outliers. A robust design should have boxes and whiskers of short length and as small a number of outliers as possible. Furthermore, a high performance behaviour should have an error-median value close to zero.

The performance of each objective is defined by the following formula.

$$e_i = |\dot{x}_i - \dot{x}_{\mathrm{T}i}| + |\dot{y}_i - \dot{y}_{\mathrm{T}i}| + C|h_i - h_{\mathrm{T}i}|. \tag{6.7}$$

Where e_i is the error measure; $\dot{x}_i$ and $\dot{y}_i$ are the CoM velocities at lift-off (from parameters 'vcm1x' and 'vcm1y') of the best behaviour for objective i; $\dot{x}_{\mathrm{T}i}$ and $\dot{y}_{\mathrm{T}i}$ are the target values for $\dot{x}_i$ and $\dot{y}_i$; h_i (from 'hcm1') is the robot's angular momentum at lift-off of the best behaviour for objective i; $h_{\mathrm{T}i}$ is the target value for angular momentum about the CoM at lift-off (is 0 for all objectives except objective number 12, which is the triple somersault); and C is the conversion factor to allow the

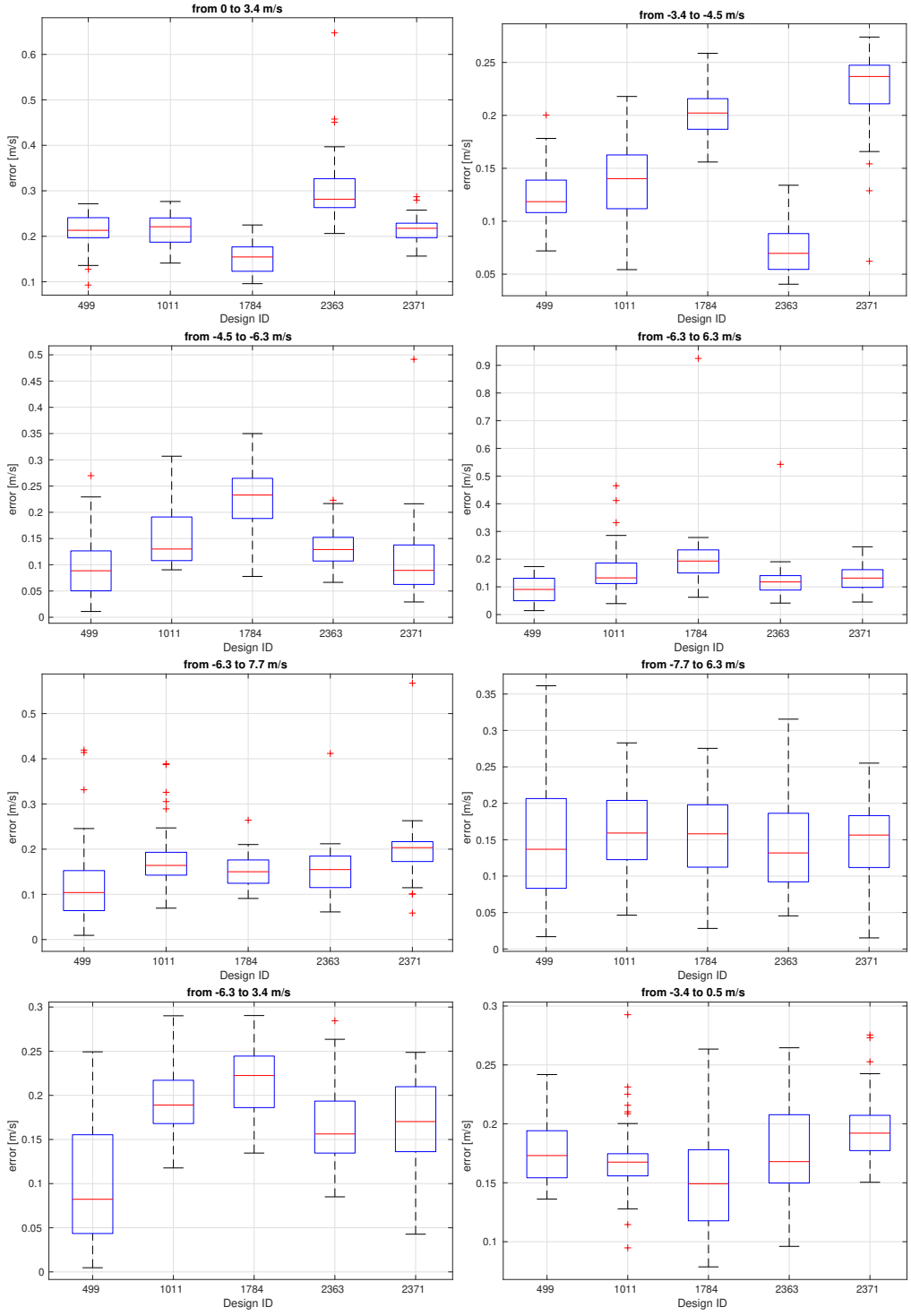

Figure 6.6: Box-whisker plots for all vertical hops (objectives 1–8) in Table 6.5. These plots summarise the results of the robustness optimisation study. In each figure, each box-whisker represents the dispersion and skewness of the 50 random designs generated around each of the five examined designs.

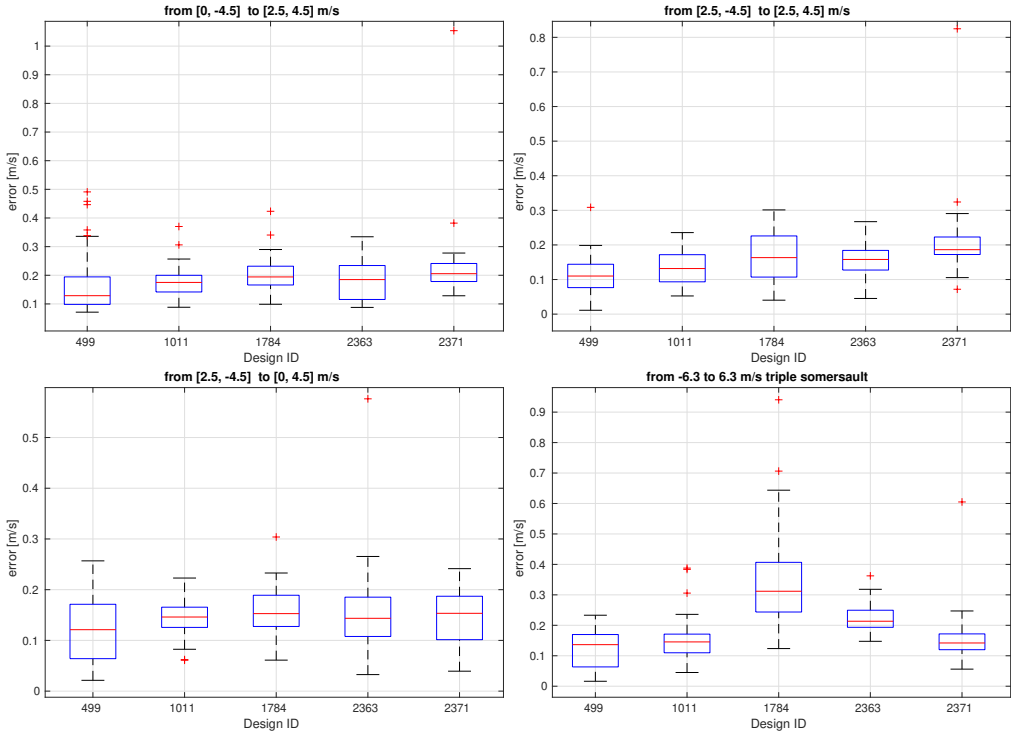

Figure 6.7: Box-whisker plots for all travelling hops (objectives 9–12) and the 2 m triple somersault (objective 12) in Table 6.5. These plots summarise the results of the robustness optimisation study.

addition of an angular momentum to a linear velocity, which is described in Section 6.6.4. In the examined case the standard deviation tolerance was set at 0.1 m/s for the desired lift-off conditions. Designs that the standard deviation (STD) of all of their objectives is below the aforementioned value are deemed successful.

From the five examined designs only one failed given the robustness evaluation criteria. Design 1784 (best hopper) has an STD value of 0.13 m/s, which violates the acceptable tolerance of 0.1 m/s. One can also easily notice from the boxplots that this particular design has the highest median error value in most of the objectives (7 out of 12) and high variability (long whiskers) in the triple somersault and the halting hop (performance objectives 8 and 12). Design 499 has in most cases the lowest median error value and low variance in most objectives, which demonstrates high robustness.

Design 1011 (best somersaulter) has an acceptable average median value and in a few cases such as in the objectives 5 and 8 (the 3 m hop and the halting hop) has many outliers, an indication of possible high variability, which is not desirable. The best balancing machine (design 2371) has also an acceptable robustness performance; however, it has a below average performance in low hops (objectives 1,2,7,8), with high median and STD values. Finally, the best traveller (design 2363) demonstrates high robustness under the selected design parameter variations with low median and STD values, and has only a few outliers.

Based on the results of the robustness analysis design 499 (best overall performance) and design 2363 (best travelling performance) are shown to be robust to random variations in the selected design parameters, which means that the real performance of the final design has a higher chance of being more consistent under these variations.

Discussion

Skippy's design and behaviours are modelled by more than 100 parameters, out of which 14 of them were examined in this analysis. Other parameters that could have a significant influence on the robot's performance are the initial conditions of each hop, which are the linear velocity and the angular momentum about the robot's CoM. In reality these values will most probably never be exactly the same as in the simulation, and might have small variations (e.g., the robot has a bit of positive angular momentum). However, evaluating them will exponentially increase the number of simulations. To address this issue, the main bottleneck of this approach must be dealt with, which is the simulation execution time, a topic that is discussed in Chapter 7.

The results of this analysis also shed light into the performance difference between the initial seed (design 0) and design 499 (best overall performance) (see Section 6.6.5), where it was shown that only a 0.02 rad difference in the ankle curvature between two different designs can lead to a non-negligible improvement in the overall performance. We now know that small acceptable variations in the examined design parameters can result in significant, but acceptable alterations in performance. Based on these results, we can expect slight variations in the robot's performance with different parts or robot modifications (e.g., when different springs of the same batch are used or by miss-calculating or changing the torso's CoM x coordinate).

Furthermore, the outcome of this experiment gives a deeper insight on the possible ramifications of design imperfections on the final performance of the robot. For example, if Skippy is built according to the specifications of design 499 and it is able to achieve only a maximum hopping height of 2.8 m (instead of 3 m) it will not be a surprise since robustness analysis has shown that this is a possible performance under imperfections in the examined design parameters, which can result in a lot less wasted time during the experimental process (e.g., trying to improve the controller for achieving an unfeasible performance).

6.6.8 Selecting the Best Design

In the previous section, it has been shown that designs 1011, 1784 and 2371 are not very robust to certain design variations, even though they meet the performance objective criteria. So the final candidates are designs 499 (best overall performance) and design 2363 (best travelling hopper). In summary,

- both designs meet robustness criteria;
- design 2363 spends 10% less energy than design 499 to achieve the required performance;

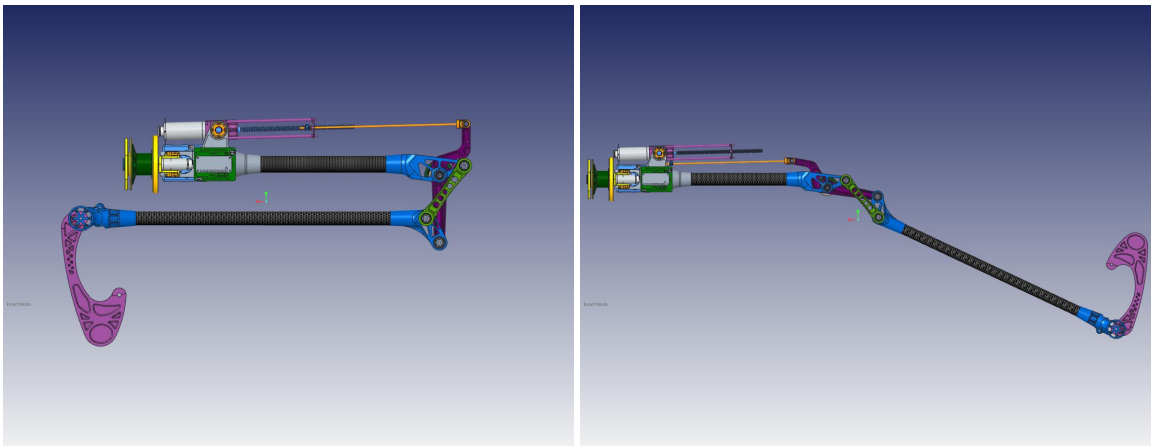

Figure 6.8: Skippy design in Creo CAD software according to the specifications of the selected design (499). Left: closed hip configuration; right extended hip configuration. The main and ankle fibreglass leaf-springs, the end stops, the ring screw nut and the crossbar are not presented.

- design 499 has a higher performance and achieved the objectives with 63% less error than that of design 2363;
- design 499 has 8.8% higher AVG value than design 2363, which makes it a much better balancing machine.

For the aforementioned reasons, design 499 is the final winner of this design optimisation study because it demonstrates the best performance-robustness trade-off. Even though energy efficiency is also considered in the final design selection process, performance is prioritised (in both balancing and hopping), and for this reason design 499 is selected over design 2363, which is a more energy efficient design.

Figure 6.8 shows an incomplete design (not yet finished) of the real Skippy according to the specifications of design 499. The crossbar, the ankle and hip fibreglass spring, the end stops and the ring screw nut are not presented. The links of the robot are made out of carbon fibre, which is a lightweight and strong material. The blue and purple parts will be made of aluminium. The two Maxon motors are visible (in grey); the top motor actuates the hip joint via the ring screw, and the bottom smaller motor will actuate the crossbar via a capstan drive. Notice that most of the mass (batteries, motors, electronics) is located at the end of the torso joint, which is called the head. Adjustment of the distribution of these parts will allow to achieve the desired torso's CoM x coordinate, an important parameter for the robot's performance.

6.6.9 Behaviours of the Best Design

In this section the optimal behaviours of design 499 are shown and several plots of various quantities of the robot during these behaviours are presented and discussed. Specifically, voltage and current profiles for all the performance objectives are presented, example plots of the ground reaction forces,

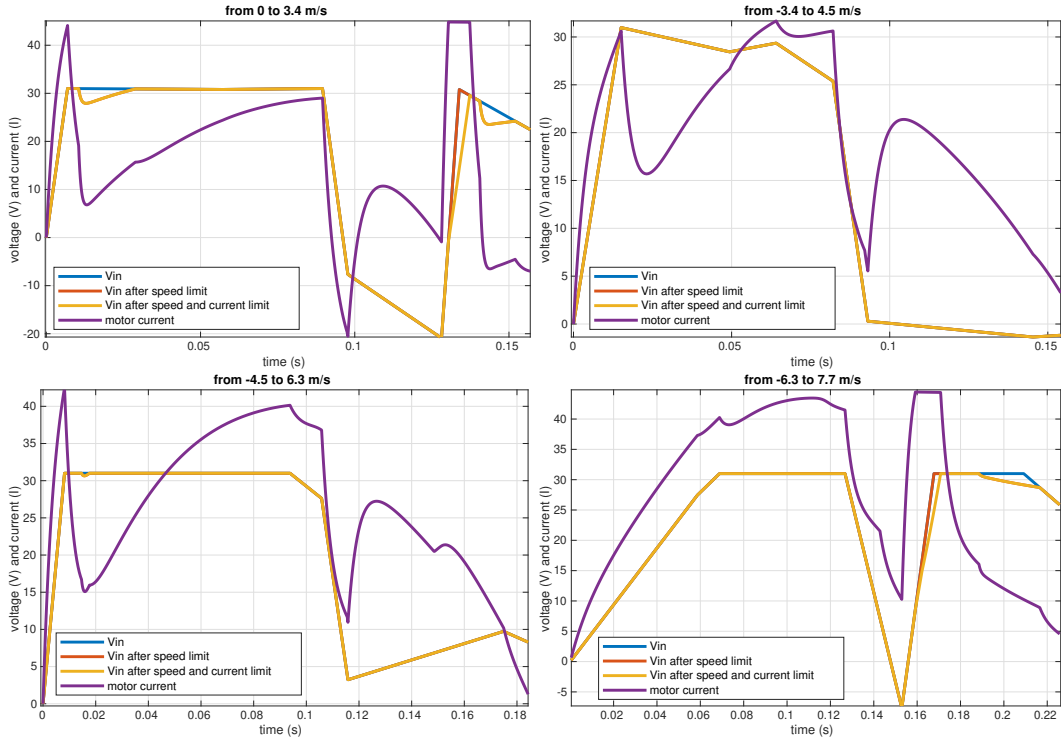

Figure 6.9: Voltage and current profiles for increasing-height vertical hops. Top left: from rest to a 0.6 m hop; top right: from a 0.6 m to a 1 m hop; bottom left: from 1 m to a 2 m hop; bottom right: from a 2 m to a 3 m hop.

angular momenta, generated torques/forces and energy flows. A visualisation of the 12 optimal behaviours of design 499 is presented in Appendix A, where motions strips (screenshots) from the simulation videos are presented.

To obtain the maximum physical performance and reach the performance envelope, the robot must operate near its limits. To do that, Skippy must saturate its motor and push the mechanism to its limits. This can indeed be seen in the voltage and current profiles of Designs 499. The 12 optimal behaviours are presented in Figures 6.9, 6.10, 6.11, and 6.12. In the aforementioned graphs, the three 'Vin' signals are drawn in the order that they appear in the legend. This means that the blue curve is visible only where speed limiting is in effect, because otherwise it is obscured by the red curve, and that the red curve is visible only where current limiting is in effect, because otherwise it is obscured by the orange curve.

In more detail, seven out of the 12 behaviours reach the ring screw's speed limit; seven out of 12 behaviours reach voltage saturation; and three behaviours reach current saturation. These results indicate that Design 499 reaches its performance envelope in most of the desired performance objectives.

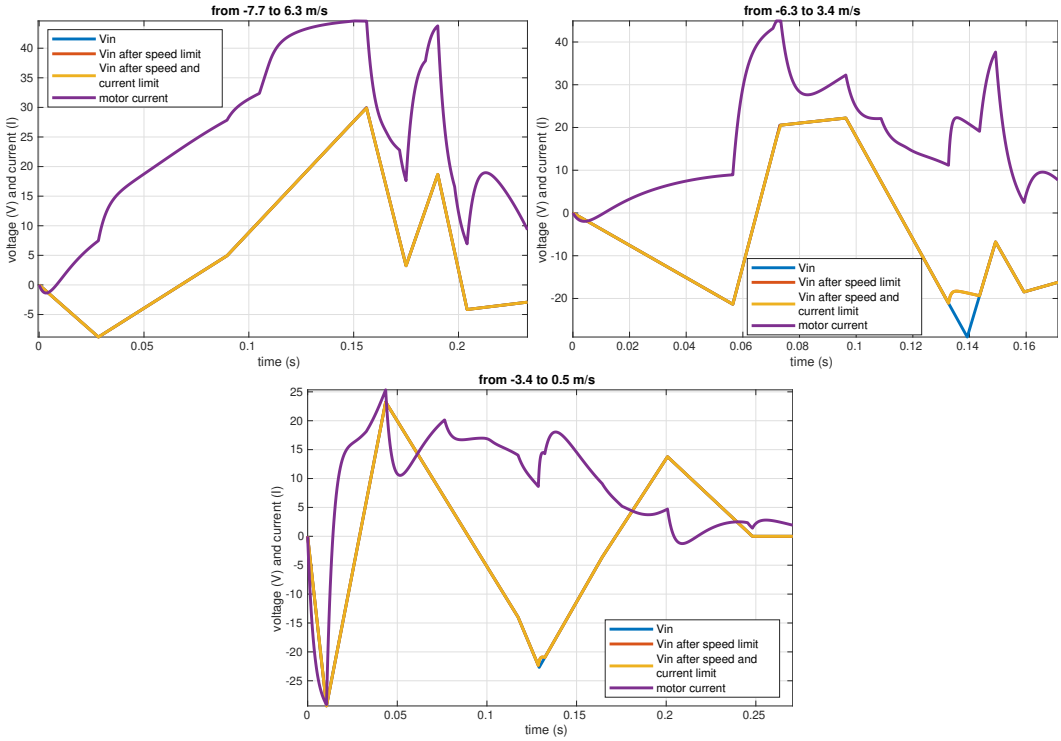

Figure 6.10: Voltage and current profiles for decreasing-height vertical hops. Top left: from a 3 m to a 2 m hop; top right: from a 2 m to a 0.6 m hop; bottom: from a 0.6 m to a 0.01 m hop.

Vertical Hops

Figure 6.9 presents all of the increasing-height hops required to reach the 3 m hop (objectives 1–4 in Table 6.5). In the 3 m hop (bottom right graph) the robot spends about 40% of its time in voltage saturation and 5% in current saturation. The effect of the ring screw's speed limit is visible at about 0.19 seconds for about 12% of the behaviour's time, showing not only the motor but the ring screw mechanism is also driven to its limits. In the initial hop from rest to a 0.6 m hop, the robot also struggles to meet the objective since it reaches the ring screw's speed limit twice, and also spends 50% of the behaviour in voltage saturation and 5% in current saturation. This extreme behaviour however cannot be kept up for a long time. In reality this will cause the motor to overheat. To prevent this, real Skippy will be equipped with sensors to monitor the temperatures of the motors, and just like any athlete, when Skippy overexerts itself it will need some time to rest. The robot can be seen to have an easier time during the two intermediate hops (top right and bottom left figures) indicating that the mechanism could perform better in these two objectives.

In addition to the voltage and current profile, the ground reaction forces (grf), and the pulling force of the nut in the 3 m hop are presented in Figure 6.13 (left figures). In both of these plots a bump can be observed. This bump is the result of the mechanism hitting its hip end stop during this behaviour. Ideally, hitting the end stops should be avoided due to the energy losses. However, results from earlier experiments have shown that a variety of strategies exist for achieving the objectives,

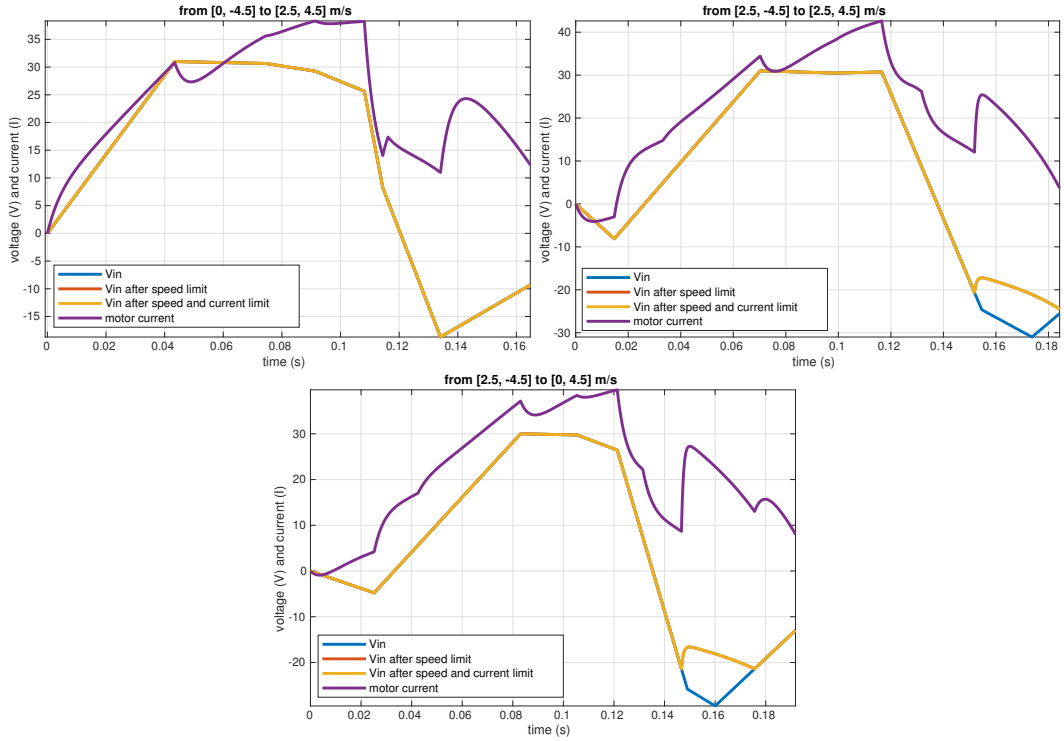

Figure 6.11: Voltage and current profiles for travelling hops. Top left: starting a travelling hop from a vertical hop of 1 m; top right: repeated travelling hop of 1 m height and 2.3 m distance; bottom: from a travelling hop to a vertical hop of 1 m.

meaning that this issue is not a decisive factor. Finally, it can also be observed that a real Skippy would slip for a very short amount of time at both the beginning and the end of both stance phases, because the tangential component of the ground-reaction force is outside the friction cone. In the force profile of Figure 6.13 (top left) one can also observe the pulling force reaching very close to the thrust bearings' limit of 1500 N, which is a further indication that the 3 m behaviour is pushing the mechanism to its limits.

Figure 6.10 shows the voltage and current profiles of the three decreasing height hops from a 3 m hop to rest. From these hops it can be easily observed that the robot does not have a hard time meeting the given performance objectives. Specifically, the ring screw speed limit is the only limit reached, but only for a short amount of time in the final two hops; the motor limits are not reached in any of these behaviours.

Travelling Hops

The voltage and current profiles of the travelling hops are presented in Figure 6.11. In these behaviours the ring screw's speed limit is reached in addition to some voltage saturation. However, despite this fact, the behaviours have a large margin from the constraint limits and no end stops are hit, meaning that there is more potential to the mechanism's travelling speed. This is expected though, because these

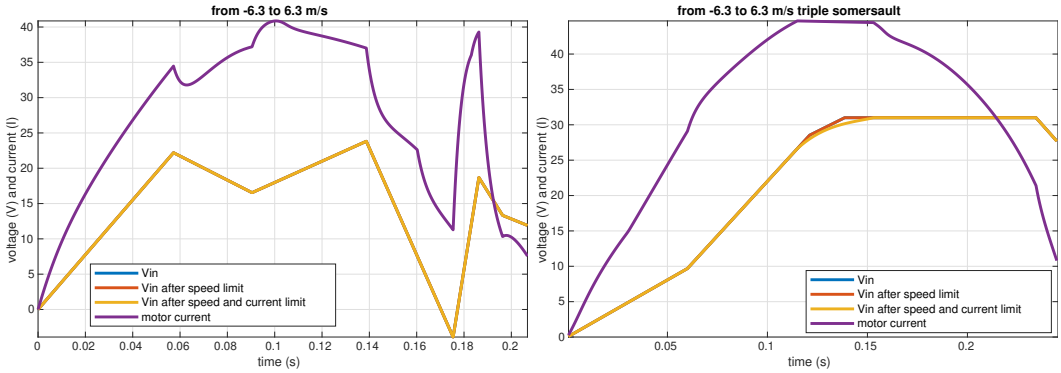

Figure 6.12: Voltage and current profiles for the 2 m repeated vertical hop (left), and the 2 m and 3.5 $\mathrm{kgm^2s^{-1}}$ triple somersault (right).

objectives were added only for versatility purposes, and not to push the mechanism to its limits. In fact, in subsequent studies, that are not presented in this thesis, mechanisms capable of reaching a travelling speed of 5.3 m/s (19 km/h) in a series of successively increasing travelling speed hops were discovered.

Repeated Hop

Figure 6.12 (left) shows the voltage profile to achieve a 2 m repeated hop with zero angular momentum and horizontal velocity. During this behaviour no limit was reached but the hip end-stop was hit. In addition, all the constraints (Section 6.2) were far from their limits. This indicates that the mechanism has more potential in this performance objective too, which could be explored in a future study.

Somersault

The 2 m triple somersault, together with the 3 m vertical hop, are the robot's most demanding behaviours. In Figure 6.12 (right) one can observe a voltage saturation during about 40% of the behaviour, and a current saturation during about 14% of the stance phase. Furthermore, the hip end stop was hit. Specifically, in Figure 6.13 (right) we can observe that Skippy hits the end stop less hard than in the corresponding behaviour for a 3 m hop. In addition, the optimal behaviour for the somersault comes close to most of the constraint limits, including the maximum thrust bearing force (top right of Figure 6.13).

Figure 6.14 shows four plots of the torques generated at the ankle joint and the hip end stop, and angular momentum about the torso's CoM for the 3 m hop (top figures) and the 2 m triple somersault (bottom figures). In the 3 m hop case the hip end-stop was hit more than two times harder than in the somersault case, which can be observed from the bumps in the left figures. An interesting observation is the zero-crossing of the ankle's joint torque, which happens in the somersault but not in the 3 m hop. As mentioned in Section 5.3.1, the spring-loaded ankle affects the direction and magnitude of the grf, and the aforementioned result shows that this behaviour requires more sophisticated control

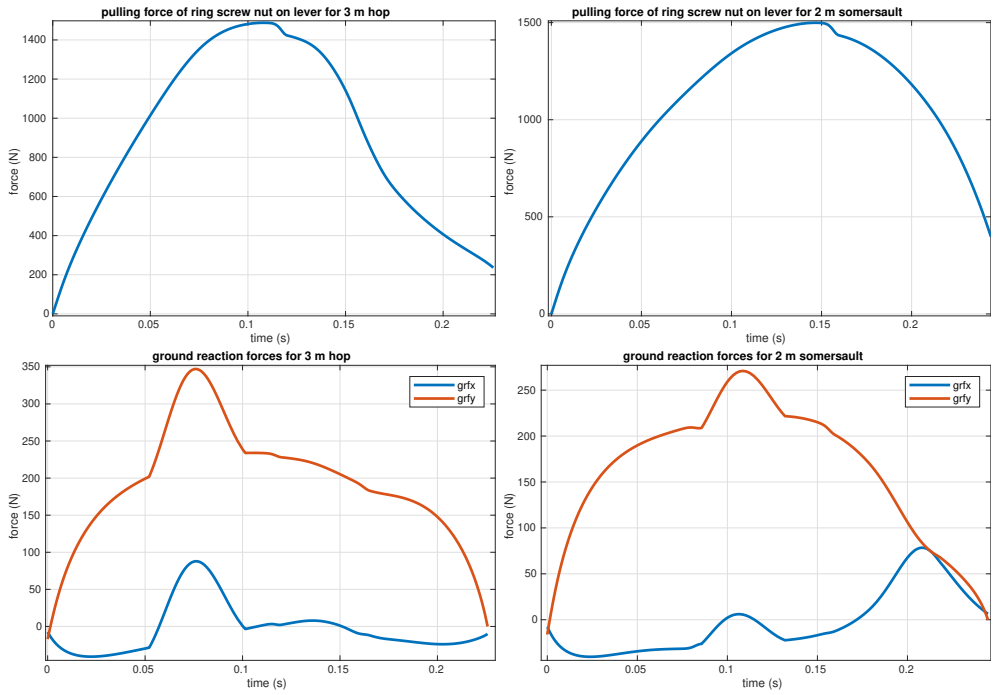

Figure 6.13: Pulling force of the ring screw nut (body B_{10} in Figure 5.4), and ground reaction forces (grf) of Skippy's most demanding behaviours. Left: pulling force (top), and grf (bottom) during the 3 m vertical hop; right: pulling force (top) and grf (bottom) during the 2 m triple somersault.

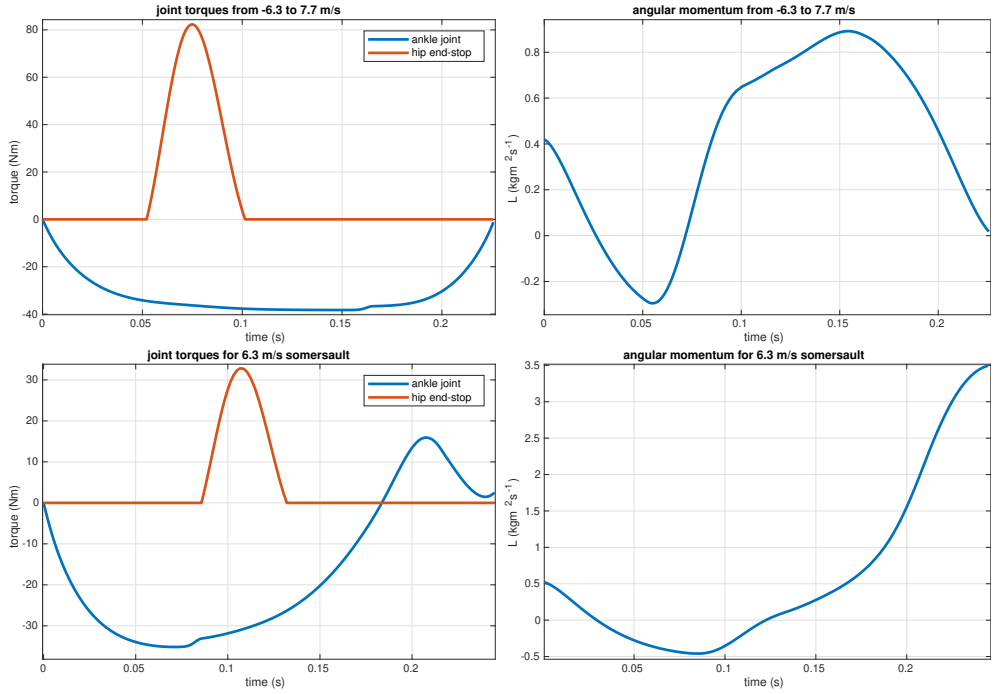

Figure 6.14: Left figures: torques produced by the ankle spring and the hip joint generated during the 3 m hop (upper) and the 2 m triple somersault (lower); right figures: angular momentum about the torso's CoM.

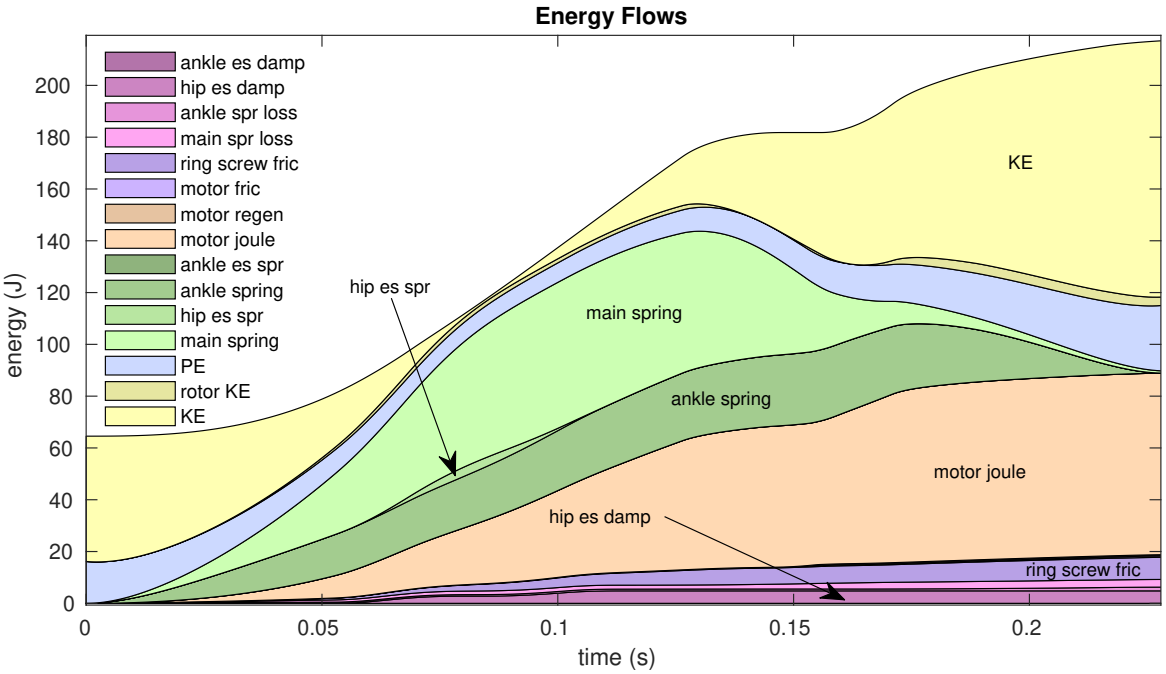

Figure 6.15: Stacked Energy flow during the 0.228 second stance phase of a 7.7 m/s (3 m) hop. The energy losses due to damping and friction can be observed from the bottom curves. During this stance phase the hip end stop was hit ('hip es damp' shows the energy lost to the damping effect, and 'hip es spr' is the stored elastic energy, which is returned to the mechanism as it bounces off the end-stop).

than the 3 m hop. Specifically, in the somersault the ankle is being extended rather than being flexed. This is caused by the leg trying to kick backwards in order to increase its angular momentum before lift-off. This effect can be observed in the bottom right figure, where one can notice the very rapid increase of angular momentum during the same time that the ankle starts extending. Moreover, in the 3 m hop (top right figure) we can observe how the robot crouches during the first milliseconds of the behaviour, and then kicks upwards to achieve the desired lift-off velocity.

Energy

A better overall view of what is happening to the system during the stance phase can be observed through the energy flows shown in Figures 6.15 and 6.16. In the energy flow of the high hop (Figure 6.15) we can observe the energy loss due to the collision with the hip end-stop, which is much less in the somersault (Figure 6.16). Not hitting the end stop means better and smoother control, which can be observed by the smoother curves produced during the somersault stance phase. The yellow (KE) and the two large green areas ('main spring' and 'ankle spring') illustrate the conversion of initial kinetic energy to stored elastic energy of the springs, back to the final kinetic energy to propel the robot for the desired objective. We can also observe the losses due to the ring screw friction and joule heating of the motor winding ('motor joule'). To give an idea of these values the kinetic energy to propel the robot into a 3 m hop is ~ 90 J and the energy lost due to heating of the motor winding is

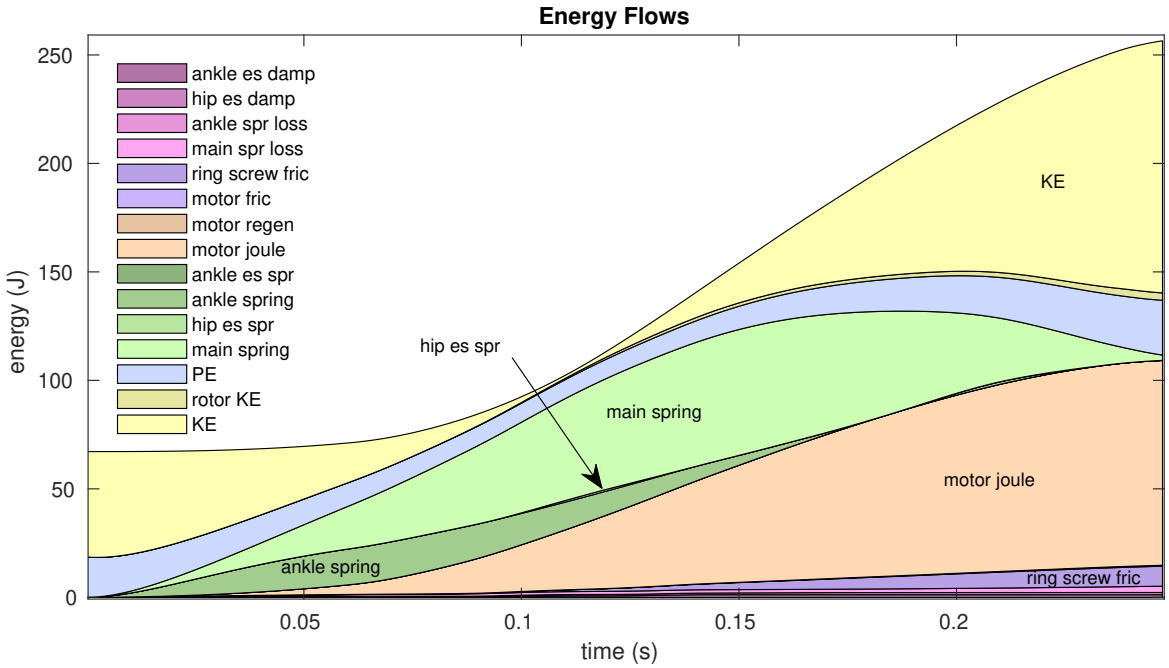

Figure 6.16: Stacked Energy flow during the 0.247 second stance phase of a 6.3 m/s (2 m) triple somersault. In this figure the energy flows are much smoother compared to Figure 6.15, which can be largely attributed to the fact that the hip end-stop was not hit very hard.

$\sim$ 70 J. This is a significant amount of lost energy and not taking it into consideration can result in discrepancies between simulation and reality.

Figure 6.17 shows the battery energy spent during the optimal behaviours for each performance objective in Table 6.5. The most energy-demanding behaviour is the 2 m triple somersault, which requires 190.2 J, followed by 153.2 J for the 3 m hop. Travelling hops require a moderate amount of energy to be spent, while the least energy is spent by the two hops to reduce the hopping height from 2 m to 0.6 m and then to rest. Interestingly, the initial hop does not follow the pattern of the other vertical hops, where energy increases/decreases with hopping height, but consumes more energy than its successive increasing height hop (the 1 m hop). This also demonstrates the fact that building up energy is a difficult task, since the springs are unloaded at the beginning of the first hop. In that particular hop the optimiser chose to straighten up, and launch with an upward momentum; however, a better strategy for an initial hop could be to straighten up a little, then crouch rapidly, and then suddenly launch into an initial hop while the CoM has significant downward momentum.

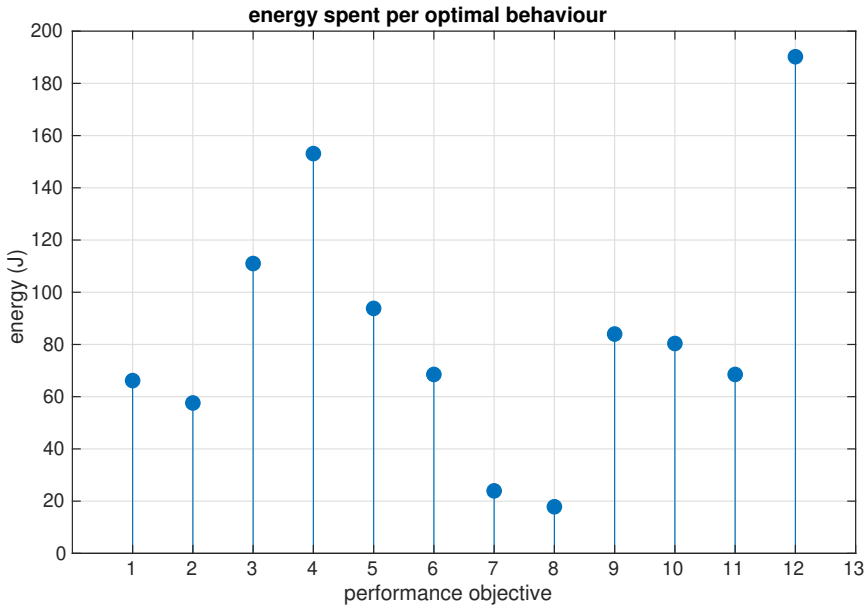

Figure 6.17: Energy spent by the batteries in each of the 12 optimal behaviours of design 499. Each point in the graph corresponds to the energy spent for the optimal behaviour to reach a performance objective (Table 6.5). Points 1–4 are the increasing height hops; point 5 is the repeated hop; points 6–8 are the decreasing height hops; points 9–11 are the travelling hops; and point 12 is the somersault.

Chapter 7

Discussion

This chapter discusses some of the limitations and possible improvements to the proposed optimisation approach and design study. Obtaining accurate models of the robot's components could be a cumbersome and challenging task; and the reason is that deep understanding of the underlying examined systems is required, and sometimes, sophisticated and expensive equipment and software. However, by doing so, one can reduce the gap between simulation and reality, which can lead to robots that demonstrate unprecedented behaviours, and in fewer design cycles.

The bottleneck of this approach is the simulation cost. For the case-study of this thesis, three different programs were used for modelling, simulating and co-optimising the design and the behaviours of Skippy. These three programs (Matlab, Simulink, and ModeFrontier) introduced large execution overheads with Simulink, being the worst offender. To be more specific, in the final optimisation experiment presented in Section 6.6, 400 Skippy designs were examined, and for each design 12 behavioural optimisation studies were performed. In each behaviour optimisation experiment, 1000 behaviours were explored (Table 6.8), so we get a total of 4,800,000 simulations. The machine used had 16 GB of RAM and used an Intel Core i7-4820k 8×3.7 GHz. The execution time for the design study was approximately 12 days, which is 1,036,800 seconds. So the software was performing approximately 4.6 simulations per second. However, a rough count of the number of arithmetic operations needed to perform one simulation suggests that we should have been getting closer to 100 simulations per second, implying that overheads accounted for something like 95% of total execution time. The single biggest overhead is Simulink's simulation initialisation phase, which takes far longer than the execution phase.

The lack of efficiency is the main reason why the presented optimisation experiments (Sections 6.5 and 6.6) had a relatively small generation length and population size. However, the choice of seven mechanism design parameters is comparable with other works in the literature. For example, Ha et al. [46] allow up to seven design parameters to vary in their study; Saar, Giardina and Iida [92] optimise four; and Spielberg et al. [104] optimise eight. Moreover, in the design study of Section 6.6, 22 parameters are being optimised, because there are also 15 behaviour parameters. As for the set of

performance objectives, the only significant omission is an objective to recover from a triple backward somersault back into a vertical hop.

Slow execution time could be mitigated by the following actions: (1) software platforms with higher efficiency can be used (i.e., less overheads), (2) software in a more efficient language can be implemented (e.g., C instead of Matlab), or (3) harness the power of cloud-computing, for extra computational power. Some, if not all of these solutions require extra resources, but are not restrictive; particularly, cloud-computing is gradually becoming more accessible due to an increase in popularity over the past years.

Although the dynamic model is highly detailed and realistic in most respects, there is one item that is not realistic: the kinematic constraint that forces the foot to roll without slipping on the ground regardless of the contact forces. One improvement that could be made to the model is to replace this constraint with a realistic model of compliant frictional contact between the robot and the ground. A compliant model is required because Skippy will have a soft rubber foot and even softer crash protection foam. This would allow to model and study the bouncing and slipping that occurs when the foot lands, as well as the slipping just before lift-off. It would also allow the study of flight-phase behaviours. Furthermore, the case-study is limited to a plane. Performing a full 3 D study will result in a more complete study and will allow to examine more complicated behaviours, such as pirouettes. Moreover, the performance envelope in other behaviours such as travelling hops could be explored. In fact, Skippy designs were discovered in later studies than the one presented in this thesis, that can reach and maintain travelling speeds of 5.3 m/s (hops of 1.7 m height and 6.26 m distance), which corresponds to 19 km/h, in a series of successive travelling hops, starting from rest.

Other directions for future work would be to apply the proposed optimisation approach to the design and behaviour co-optimisation of other types of robots, such as soft robots. Finally, an important step would be to validate the effectiveness of the proposed approach by building Skippy, and demonstrating the behaviours that it is designed to perform.

Chapter 8

Conclusion

This thesis presented an approach for design and behaviour co-optimisation of mobile robots. It was divided into two parts. The first part presents a 2-layer optimisation framework that seeks the best design (Layer 1), and the optimal behaviours to achieve a set of performance requirements (Layer 2); and a proposed methodology to perform the optimisation experiments. In the second part, the optimisation philosophy is applied to a specific example, which is a high-performance balancing and hopping monopedal robot.

The proposed optimisation approach is based on realistic models of the robot's hardware and limitations, and its behaviours. As a result, the gap between simulation and reality is reduced, and the study of designs that can utilise the current technology safely and towards its maximum potential can be performed. The optimisation framework is designed so that the inherent relationship between a robot and its performance requirements is exploited via the co-optimisation of design and behaviour. The proposed optimisation methodology aims to facilitate the robot design process by: (1) identifying the most critical components and parameters of the robot's design, (2) obtaining optimal designs by maximising the qualitative information of the design-space (i.e., use the minimum amount of resources), (3) apply qualitative evaluation criteria for the selection of the design with the best trade-off between the performance requirements, and (4) meet realistic design criteria, that will allow the final design to be manufactured according to available resources and be robust to imperfections of the real world (e.g., manufacturing errors).

The effectiveness of the optimisation approach was shown via a case study on a monopedal hopping and balancing robot. In this study, the robot's performance envelope in vertical hops was sought, which includes a 3 m hop, in addition to several other demanding and conflicting performance requirements, which are: travelling hops, a high balancing ability, and a 2 m triple somersault. The final result is a design that will be built, and can theoretically achieve a large set of demanding performance objectives. This design has been shown in simulations to be able to reach its hardware's maximum potential safely, and was selected according to performance and robustness criteria. In particular, the latter criterion means that the design's performance will not be significantly deteriorated in the presence of expected uncertainties in the real world. Furthermore, with the proposed methodology

important design parameters of the examined agile monopedal robot and their effects/trade-offs on its performance were identified, which can give insights in the design process of legged robots.

The aim of this research is to show that a systematic robot design optimisation study can lead to versatile robots that display impressive behaviours with fewer design iterations. In comparison to the state of the art and other studies in the literature, this work examines the effect of multiple realistic components and limitations, that have been largely neglected, and investigates designs that are capable of achieving multiple demanding behaviours, which conflict with each other. These aspects have not been properly investigated in previous works, and the results of this research indicate that they should not be neglected, and play a crucial role for designing mobile robots that are robust, versatile, and can achieve high-physical performance.

The robot examined in the case study of this thesis has no obvious practical use; however, the findings of this thesis demonstrate that the proposed optimisation approach can be used for the design of highly capable robots that can achieve a plethora of behaviours, which can allow them to assist and support humans in various aspects of their life. The aforementioned findings suggest that a new approach for designing mobile robots can be arrived at.

Bibliography

[1] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[2] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[3] Morteza Azad. *Balancing and hopping motion control algorithms for an under-actuated robot*. PhD thesis, Australian National University, 2014.

[4] Morteza Azad and Roy Featherstone. Balancing control algorithm for a 3d under-actuated robot. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3233–3238. IEEE, 2014.

[5] Zachary Batts, Joohyung Kim, and Katsu Yamane. Untethered one-legged hopping in 3d using linear elastic actuator in parallel (leap). In *International Symposium on Experimental Robotics*, pages 103–112. Springer, 2016.

[6] Cenk Baykal and Ron Alterovitz. Asymptotically optimal design of piecewise cylindrical robots using motion planning. In *Robotics: Science and Systems, Cambridge, Massachusetts*, 2017. http://rss2017.lids.mit.edu/program/papers/08/.

[7] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009.

[8] Lionel Birglen and Thomas Schlicht. A statistical review of industrial robotic grippers. *Robotics and Computer-Integrated Manufacturing*, 49:88–97, 2018.

[9] Gerardo Bledt, Matthew J Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2245–2252. IEEE, 2018.

[10] Josh Bongard, Victor Zykov, and Hod Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[11] Josh C Bongard. Evolutionary robotics. *Communications of the ACM*, 56(8):74–83, 2013.

[12] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[13] Marco Ceccarelli and Chiara Lanni. A multi-objective optimum design of general 3r manipulators for prescribed workspace limits. *Mechanism and Machine Theory*, 39(2):119–132, 2004.

[14] Damien Chablat, Swaminath Venkateswaran, and Frédéric Boyer. Mechanical design optimization of a piping inspection robot. *Procedia CIRP*, 70:307–312, 2018.

[15] Tianjian Chen, Zhanpeng He, and Matei Ciocarlie. Hardware as policy: Mechanical and computational co-optimization using deep reinforcement learning. *arXiv preprint arXiv:2008.04460*, 2020.

[16] Nicholas Cheney, Jeff Clune, and Hod Lipson. Evolved electrophysiological soft robots. In *Artificial Life Conference Proceedings 14*, pages 222–229. MIT Press, 2014.

[17] Nick Cheney, Josh Bongard, Vytas SunSpiral, and Hod Lipson. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface*, 15(143):20170937, 2018.

[18] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

[19] Francesco Corucci, Marcello Calisti, Helmut Hauser, and Cecilia Laschi. Novelty-based evolutionary design of morphing underwater robots. In *Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation*, pages 145–152, 2015.

[20] Rafael Stanley Núñez Cruz and Juan Manuel Ibarra Zannatha. Efficient mechanical design and limit cycle stability for a humanoid robot: An application of genetic algorithms. *Neurocomputing*, 233:72–80, 2017.

[21] Michael H Dickinson, Claire T Farley, Robert J Full, MAR Koehl, Rodger Kram, and Steven Lehman. How animals move: an integrative view. *science*, 288(5463):100–106, 2000.

[22] Krishnamanaswi M Digumarti, Christian Gehring, Stelian Coros, J Hwangbo, and Roland Siegwart. Concurrent optimization of mechanical design and locomotion control of a legged robot. In *Mobile Service Robotics*, pages 315–323. World Scientific, Poznan, Poland, July 2014.

[23] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[24] Josephus J. M. Driessen, Antonios E. Gkikakis, Roy Featherstone, and B. Roodra P. Singh. Experimental demonstration of high-performance robotic balancing. In *2019 International Conference on Robotics and Automation (ICRA)/IEEE*, pages 9459–9465, Montreal, Canada, 2019.

[25] Boston Dynamics. Spot robot, 2021. https://shop.bostondynamics.com/spot, accessed Apr. 2021.

[26] Elmo. Gold Twitter, 2021. https://www.elmomc.com/product/gold-twitter/, accessed Apr. 2021.

[27] Esteco. ModeFrontier2018R3, 2018. https://www.esteco.com/modefrontier, accessed Apr. 2021.

[28] Anthony C. Fang and Nancy S. Pollard. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics (TOG)*, 22(3):417–426, 2003.

[29] Roy Featherstone. *Rigid body dynamics algorithms*. Springer, New York, 2008.

[30] Roy Featherstone. Quantitative measures of a robot's physical ability to balance. *The International Journal of Robotics Research*, 35(14):1681–1696, 2016.

[31] Roy Featherstone. A simple model of balancing in the plane and a simple preview balance controller. *The International Journal of Robotics Research*, 36(13-14):1489–1507, 2017.

[32] Roy Featherstone. Ringscrew project webpage, 2021. http://royfeatherstone.org/ringscrew, accessed Apr. 2021.

[33] Roy Featherstone. Skippy project webpage, 2021. http://royfeatherstone.org/skippy, accessed Apr. 2021.

[34] Roy Featherstone. Spatial V2 Library, 2021. http://royfeatherstone.org/spatial/index.html, accessed Apr. 2021.

[35] Jacob Fraden. *Handbook of modern sensors*, volume 3. Springer, 2010.

[36] Yang Gao and Steve Chien. Review on space robotics: Toward top-level science through space exploration. *Science Robotics*, 2(7), 2017.

[37] Moritz Geilinger, Roi Poranne, Ruta Desai, Bernhard Thomaszewski, and Stelian Coros. Skaterbots: Optimization-based design and motion synthesis for robotic creatures with legs and wheels. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[38] Antonios E Gkikakis and Roy Featherstone. Realistic mechanism and behaviour co-design of a one legged hopping robot. In *2021 International Conference on Computer, Control and Robotics (ICCCR)*, pages 42–49. IEEE, 2021.

[39] Carlos Gonzalez, Victor Barasuol, Marco Frigerio, Roy Featherstone, Darwin G Caldwell, and Claudio Semini. Line walking and balancing for legged robots with point feet. pages 3649–3656, 2020.

[40] Knut Graichen, Sebastian Hentzelt, Alexander Hildebrandt, Nadine Kärcher, Nina Gaißert, and Elias Knubben. Control design for a bionic kangaroo. *Control Engineering Practice*, 42:106–117, 2015.

[41] Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, Vincent Berenz, Steve Heim, Felix Widmaier, Thomas Flayols, et al. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5(2):3650–3657, 2020.

[42] Chong Gu. *Smoothing spline ANOVA models*, volume 297. Springer Science & Business Media, 2013.

[43] Sameer Gupta and Ekta Singla. Evolutionary robotics in two decades: A review. *Sadhana*, 40(4):1169–1184, 2015.

[44] Sehoon Ha, Stelian Coros, Alexander Alspach, James M Bern, Joohyung Kim, and Katsu Yamane. Computational design of robotic devices from high-level motion specifications. *IEEE Transactions on Robotics*, 34(5):1240–1251, 2018.

[45] Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. Task-based limb optimization for legged robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2062–2068. IEEE, 2016.

[46] Sehoon Ha, Stelian Coros, Alexander Alspach, Joohyung Kim, and Katsu Yamane. Computational co-optimization of design parameters and motion trajectories for robotic systems. *The International Journal of Robotics Research*, 37(13-14):1521–1536, 2018.

[47] Harmonic Drive. Harmonic Drive, 2021. https://www.harmonicdrive.net/, accessed Apr. 2021.

[48] Christopher Hazard, Nancy Pollard, and Stelian Coros. Automated design of robotic hands for in-hand manipulation tasks. *International Journal of Humanoid Robotics*, 17(01):1950029, 2020.

[49] Elco Heijmink. The ring screw: Modelling, development and evaluation of a novel screw transmission. Master's thesis, TU Delft Mechanical, Maritime and Materials Engineering, 2018. http://resolver.tudelft.nl/uuid:e7e28551-0410-4303-a10a-92ec88aa91a4.

[50] Jonathan Hiller and Hod Lipson. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466, 2011.

[51] Toby Howison, Simon Hauser, Josie Hughes, and Fumiya Iida. Reality-assisted evolution of soft robots through large-scale physical experimentation: a review. *Artificial Life*, 26(4):484–506, 2021.

[52] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44. IEEE, 2016.

[53] Clay Mathematics Institute. Millenium prize problems, 2021. https://en.wikipedia.org/wiki/Millennium_Prize_Problems, accessed Apr. 2021.

[54] Wei Ji and Lihui Wang. Industrial robotic machining: a review. *The International Journal of Advanced Manufacturing Technology*, 103(1):1239–1255, 2019.

[55] Navvab Kashiri, Lorenzo Baccelliere, Luca Muratore, Arturo Laurenzi, Zeyu Ren, Enrico Mingo Hoffman, Malgorzata Kamedula, Giuseppe Francesco Rigano, Jorn Malzahn, Stefano Cordasco, et al. Centauro: A hybrid locomotion and high power resilient manipulation platform. *IEEE Robotics and Automation Letters*, 4(2):1595–1602, 2019.

[56] Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301. IEEE, 2019.

[57] Gavin Kenneally, Avik De, and Daniel E Koditschek. Design principles for a family of direct-drive legged robots. *IEEE Robotics and Automation Letters*, 1(2):900–907, 2016.

[58] Sam Kriegman, Nick Cheney, and Josh Bongard. How morphological development can guide evolution. *Scientific reports*, 8(1):13934, 2018.

[59] Daniel Kuehn, Felix Bernhard, Armin Burchardt, Moritz Schilling, Tobias Stark, Martin Zenzes, and Frank Kirchner. Distributed computation in a quadrupedal robotic system. *International Journal of Advanced Robotic Systems*, 11(7):110, 2014.

[60] Averill M Law, W David Kelton, and W David Kelton. *Simulation modeling and analysis*, volume 3. McGraw-Hill New York, 2000.

[61] Chris Leger et al. *Automated synthesis and optimization of robot configurations: an evolutionary approach*. PhD thesis, The Robotics Institute, Carnegie Mellon University USA, 1999.

[62] Fei Li, Weiting Liu, Xin Fu, Gabriella Bonsignori, Umberto Scarfogliero, Cesare Stefanini, and Paolo Dario. Jumping like an insect: Design and dynamic optimization of a jumping mini robot based on bio-mimetic inspiration. *Mechatronics*, 22(2):167–176, 2012.

[63] Thomas Liao, Grant Wang, Brian Yang, Rene Lee, Kristofer Pister, Sergey Levine, and Roberto Calandra. Data-efficient learning of morphology and controller for a microrobot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2488–2494. IEEE, 2019.

[64] Hod Lipson and Jordan B Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978, 2000.

[65] Kevin Sebastian Luck, Joseph Campbell, Michael Andrew Jansen, Daniel M Aukes, and Heni Ben Amor. From the lab to the desert: Fast prototyping and learning of robot locomotion. *arXiv preprint arXiv:1706.01977*, 2017.

[66] Adriano Macchietto, Victor Zordan, and Christian R Shelton. Momentum control for balance. In *ACM SIGGRAPH 2009 papers*, pages 1–8. 2009.

[67] Mathworks. Matlab, 2021. https://uk.mathworks.com/, accessed Apr. 2021.

[68] Mathworks. Matlab Optimization Toolbox, 2021. https://uk.mathworks.com/products/optimization.html, accessed Apr. 2021.

[69] Mathworks. Simulink, 2021. https://www.mathworks.com/products/simulink.html, accessed Apr. 2021.

[70] Maxon Group. DCX32L 24V, 2021. https://www.maxongroup.com/, accessed Apr. 2021.

[71] Jared M Moore and Philip K McKinley. Evolution of an amphibious robot with passive joints. In *2013 IEEE Congress on Evolutionary Computation*, pages 1443–1450. IEEE, 2013.

[72] George P Moustris, Savvas C Hiridis, Kyriakos M Deliparaschos, and Konstantinos M Konstantinidis. Evolution of autonomous and semi-autonomous robotic surgical systems: a review of the literature. *The international journal of medical robotics and computer assisted surgery*, 7(4):375–392, 2011.

[73] Michael P Murphy, Aaron Saunders, Cassie Moreira, Alfred A Rizzi, and Marc Raibert. The littledog robot. *The International Journal of Robotics Research*, 30(2):145–149, 2011.

[74] Quan Nguyen, Matthew J Powell, Benjamin Katz, Jared Di Carlo, and Sangbae Kim. Optimized jumping on the mit cheetah 3 robot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7448–7454. IEEE, 2019.

[75] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[76] Tønnes F. Nygaard, Charles P. Martin, Eivind Samuelsen, Jim Torresen, and Kyrre Glette. Real-world evolution adapts robot morphology and control to hardware limitations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 125–132. ACM, 2018.

[77] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Books on Computer Science. Dover Publications, New York, United States, 1998.

[78] Deepak Pathak, Chris Lu, Trevor Darrell, Phillip Isola, and Alexei A Efros. Learning to control self-assembling morphologies: a study of generalization via modularity. *arXiv preprint arXiv:1902.05546*, 2019.

[79] Mark M Plecnik, Duncan W Haldane, Justin K Yim, and Ronald S Fearing. Design exploration and kinematic tuning of a power modulating jumping monopod. *Journal of Mechanisms and Robotics*, 9(1), 2017.

[80] Silvia Poles. Moga-ii an improved multi-objective genetic algorithm. 2003. proprietary document, access restricted to modeFrontier licence holders.

[81] Pololu. G2 High-Power Motor Driver 24v21, 2021. https://www.pololu.com/product/2995, accessed Apr. 2021.

[82] Shanker G. R. Prabhu, Richard C. Seals, Peter J. Kyberd, and Jodie C. Wetherall. A survey on evolutionary-aided design in robotics. *Robotica*, 36(12):1804–1821, 2018.

[83] Marc H Raibert. *Legged robots that balance*. MIT press, 1986.

[84] Agility Robotics. Agility robotics, 2021. https://www.agilityrobotics.com/, accessed Apr. 2021.

[85] Ghost Robotics. Ghost robotics, 2021. https://www.ghostrobotics.io/, accessed Apr. 2021.

[86] Unitree Robotics. Unitree robotics, 2021. https://www.unitree.com/, accessed Apr. 2021.

[87] Wesley Roozing, Zeyu Ren, and Nikos G Tsagarakis. An efficient leg with series–parallel and biarticular compliant actuation: design optimization, modeling, and control of the eleg. *The International Journal of Robotics Research*, page 0278364919893762, 2019.

[88] Andre Rosendo, Marco Von Atzigen, and Fumiya Iida. The trade-off between morphology and control in the co-optimized design of robots. *PloS one*, 12(10):e0186107, 2017.

[89] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2):1729881419839596, 2019.

[90] Daniela Rus and Michael T Tolley. Design, fabrication and control of soft robots. *Nature*, 521(7553):467–475, 2015.

[91] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach, 4th edition. 2020.

[92] Kaur Aare Saar, Fabio Giardina, and Fumiya Iida. Model-free design optimization of a hopping robot and its comparison with a human designer. *IEEE Robotics and Automation Letters*, 3(2):1245–1251, 2018.

[93] Nikolaos V Sahinidis. Mixed-integer nonlinear programming 2018. *Optimization and Engineering*, 20:301–306, 2019.

[94] Andrea Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global sensitivity analysis: the primer*. John Wiley & Sons, 2008.

[95] U. Scarfogliero, C. Stefanini, and P. Dario. Design and development of the long-jumping "grillo" mini robot. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 467–472, 2007.

[96] Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)/IEEE*, pages 9798–9805, 2019.

[97] Daniel Seidel, Dominic Lakatos, and Alin Albu-Schäffer. Data-driven discrete planning for targeted hopping of compliantly actuated robotic legs. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2261–2266. IEEE, 2018.

[98] Claudio Semini, Victor Barasuol, Jake Goldsmith, Marco Frigerio, Michele Focchi, Yifu Gao, and Darwin G Caldwell. Design of the hydraulically actuated, torque-controlled quadruped robot hyq2max. *IEEE/ASME Transactions on Mechatronics*, 22(2):635–646, 2016.

[99] Abdulaziz Shamsah, Avik De, and Daniel E Koditschek. Analytically-guided design of a tailed bipedal hopping robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2237–2244. IEEE, 2018.

[100] Freddy Sichting, Nicholas B Holowka, Oliver B Hansen, and Daniel E Lieberman. Effect of the upward curvature of toe springs on walking biomechanics in humans. *Scientific Reports*, 10(1):1–11, 2020.

[101] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, New York, USA, 1994. ACM.

[102] Bajwa Roodra Pratap Singh and Roy Featherstone. Mechanical shock propagation reduction in robot legs. *IEEE Robotics and Automation Letters*, 5(2):1183–1190, 2020.

[103] Sony. AIBO robot, 2021. https://us.aibo.com/, accessed Apr. 2021.

[104] Andrew Spielberg, Brandon Araki, Cynthia Sung, Russ Tedrake, and Daniela Rus. Functional co-optimization of articulated robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5035–5042. IEEE, 2017.

[105] Alexander Spröwitz, Alexandre Tuleu, Massimo Vespignani, Mostafa Ajallooeian, Emilie Badri, and Auke Jan Ijspeert. Towards dynamic trot gait locomotion: Design, control, and experiments with cheetah-cub, a compliant quadruped robot. *The International Journal of Robotics Research*, 32(8):932–950, 2013.

[106] Vectornav. VN-100 IMU/AHRS, 2021. https://www.vectornav.com/products/vn-100, accessed Apr. 2021.

[107] Tom Verstraten, Raphaël Furnémont, Philipp Beckerle, Bram Vanderborght, and Dirk Lefeber. A hopping robot driven by a series elastic dual-motor actuator. *IEEE Robotics and Automation Letters*, 4(3):2310–2316, 2019.

[108] Vuk Vujovic, Andre Rosendo, Luzius Brodbeck, and Fumiya Iida. Evolutionary developmental robotics: Improving morphology and control of physical robots. *Artificial life*, 23(2):169–185, 2017.

[109] Robert J Webster III and Bryan A Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 29(13):1661–1683, 2010.

[110] Alan FT Winfield and Jon Timmis. Evolvable robot hardware. In *Evolvable Hardware*, pages 331–348. Springer, 2015.

[111] Justin K Yim, Bajwa Roodra Pratap Singh, Eric K Wang, Roy Featherstone, and Ronald S Fearing. Precision robotic leaping and landing using stance-phase balance. *IEEE Robotics and Automation Letters*, 5(2):3422–3429, 2020.

[112] Allan Zhao, Jie Xu, Mina Konaković-Luković, Josephine Hughes, Andrew Spielberg, Daniela Rus, and Wojciech Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

Appendix A

In this part visualisations of the stance phase of the selected design (499) (see Section 6.6.9) are presented so the reader can get a better understanding of the behaviours that the robot is trying to achieve. For each performance objective, its optimal behaviour is simulated and rendered using the 'showmotion' graphics display of the 'spatial_V2' Matlab library [34]. Each strip displays five frames starting right after the plastic collision of the foot with the ground, and ends at the moment the foot lifts off the ground. In the motion strips the reader can see details about the various optimal behaviours including their duration, initial joint positions (hip and ankle) and final lift-off configuration. The animation does not show details of the mechanism such as springs, electronics motors, or the ring screw nut; however, the kinematics (including the 4-bar linkage) are accurate. In addition, the CoM of the torso (the blue body) is rendered as a blue sphere on the body.

One can observe that in all vertical hops, and the transition from the travelling hop to a vertical hop (Figures A.1–A.8 and A.11), that the torso's CoM is very close to the foot contact point with the ground during lift-off, which is not the case in the initial travelling hop (Figure A.9), the continuous travelling hop (Figure A.10), and the somersault (Figure A.12).

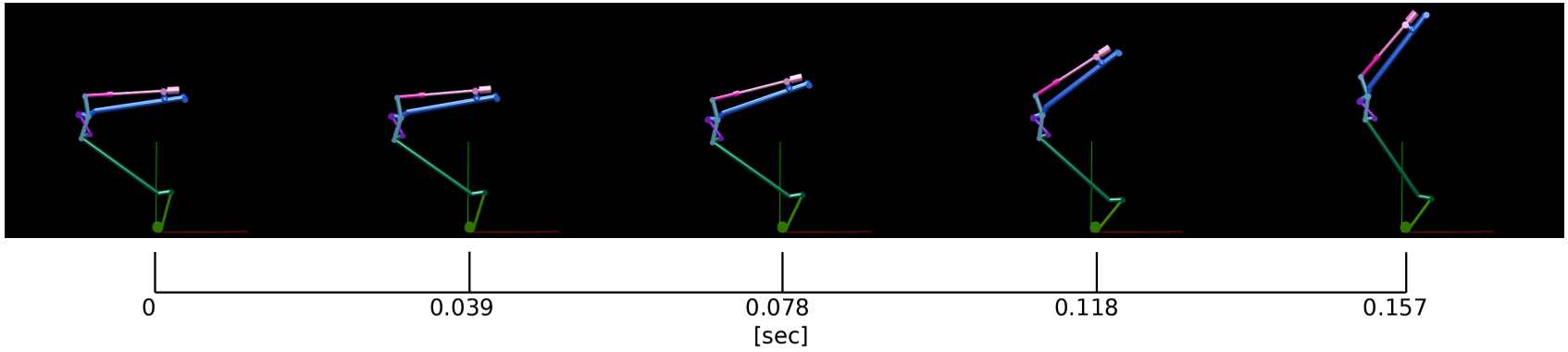

Figure A.1: Best design: Skippy stance phase simulation from 0 to 3.4 m/s].

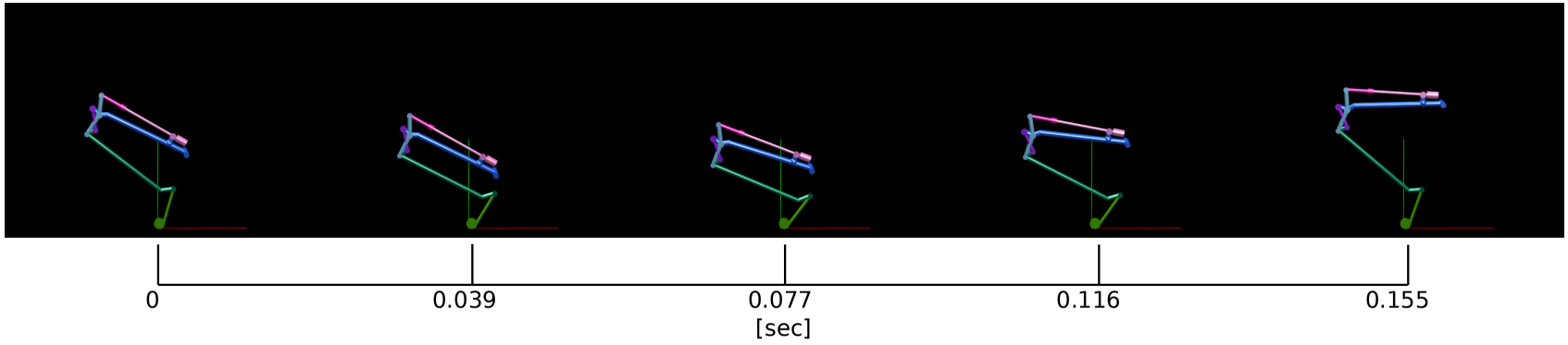

Figure A.2: Best design: Skippy stance phase simulation from −3.4 to 4.5 m/s.

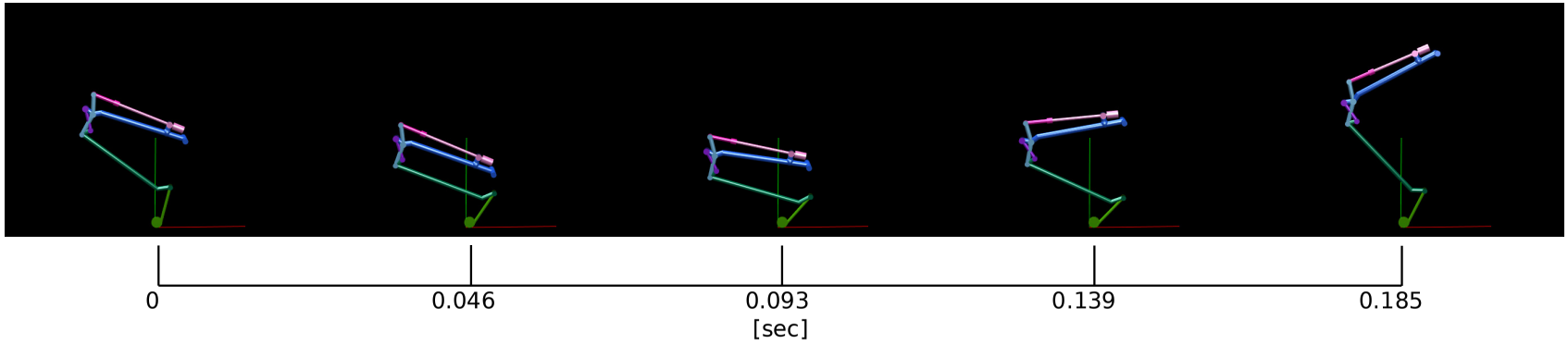

Figure A.3: Best design: Skippy stance phase simulation from −4.5 to 6.3 m/s.

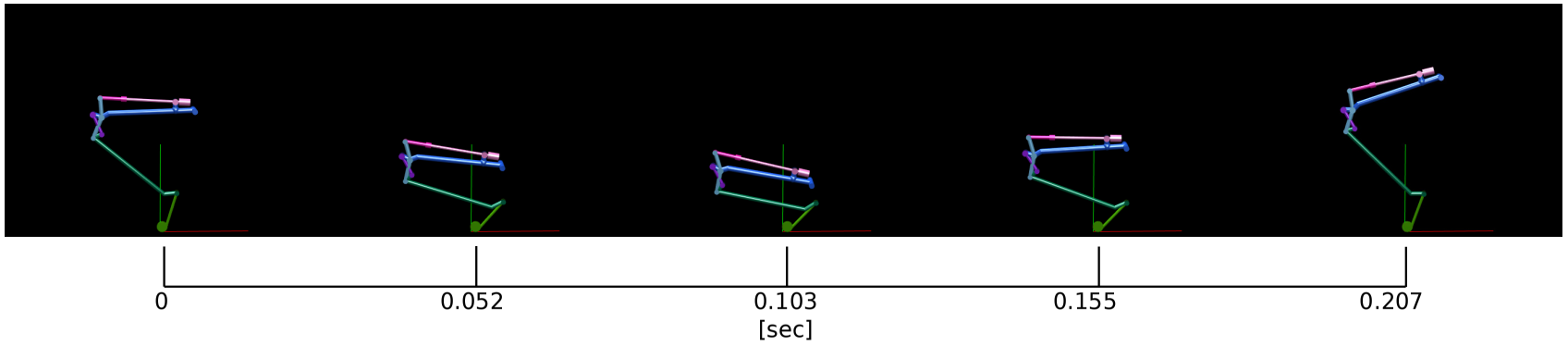

Figure A.4: Best design: Skippy stance phase simulation from −6.3 to 6.3 m/s.

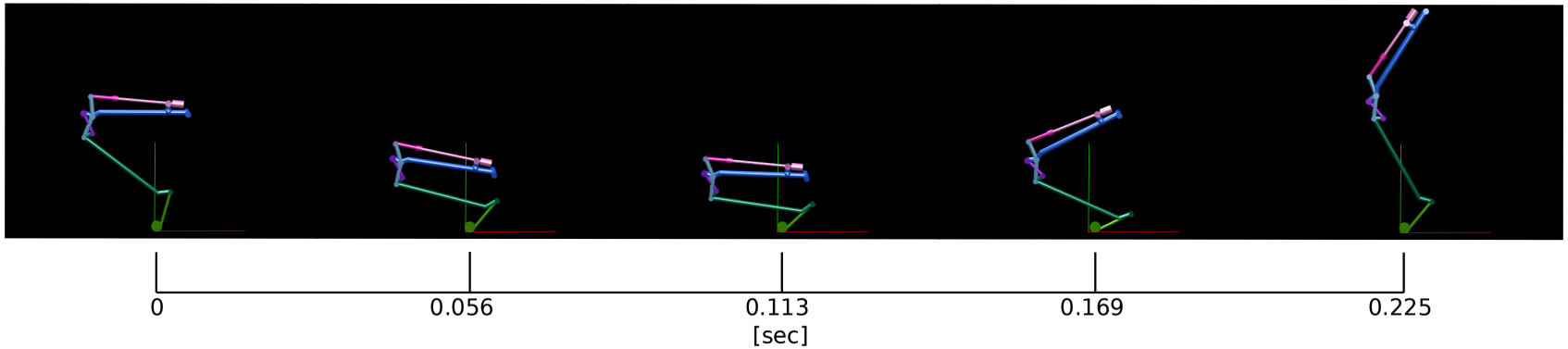

Figure A.5: Best design: Skippy stance phase simulation from −6.3 to 7.7 m/s

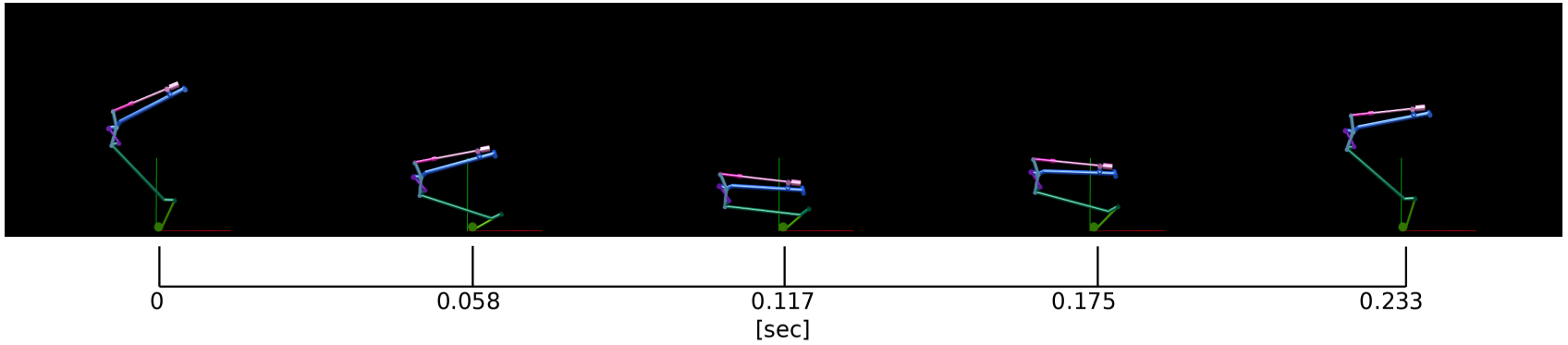

Figure A.6: Best design: Skippy stance phase simulation −7.7 to 6.3 m/s.

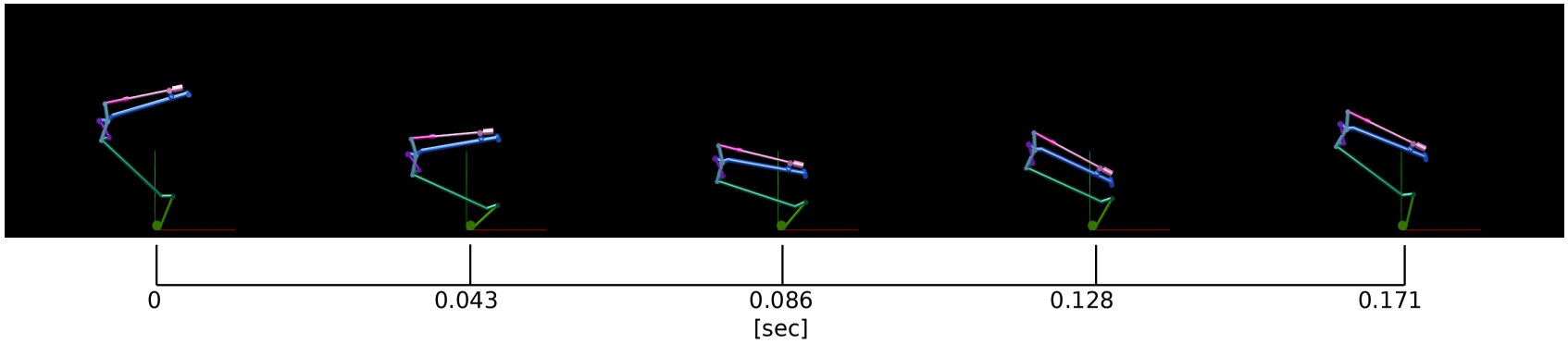

Figure A.7: Best design: Skippy stance phase simulation from -6.3 to $3.4\,\mathrm{m/s}$.

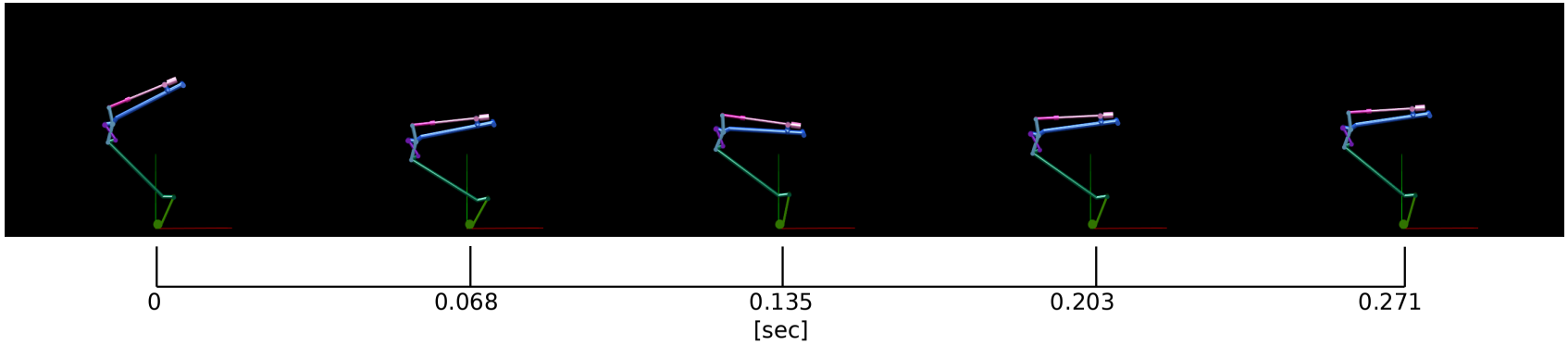

Figure A.8: Best design: Skippy stance phase simulation from -3.4 to $0.5\,\mathrm{m/s}$.

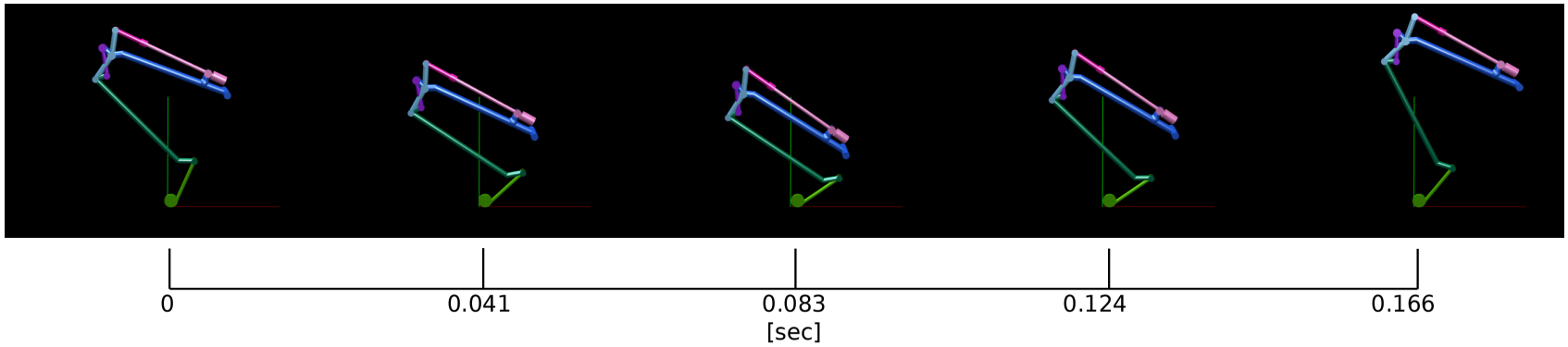

Figure A.9: Best design: Skippy stance phase simulation from $[0, -4.5]$ to $[2.5, 4.5]\,\mathrm{m/s}$.

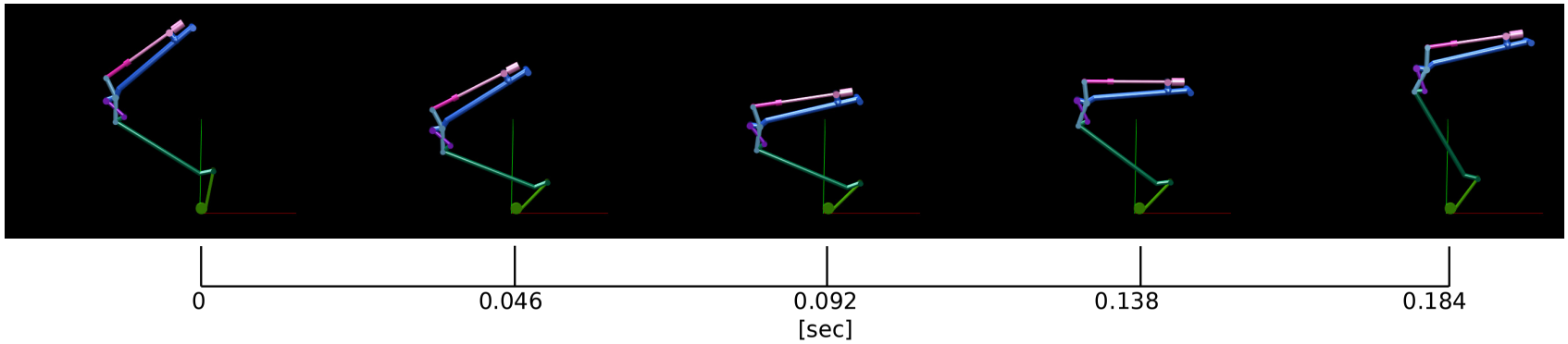

Figure A.10: Best design: Skippy stance phase simulation from $[2.5,-4.5]$ to $[2.5,4.5]$ m/s.

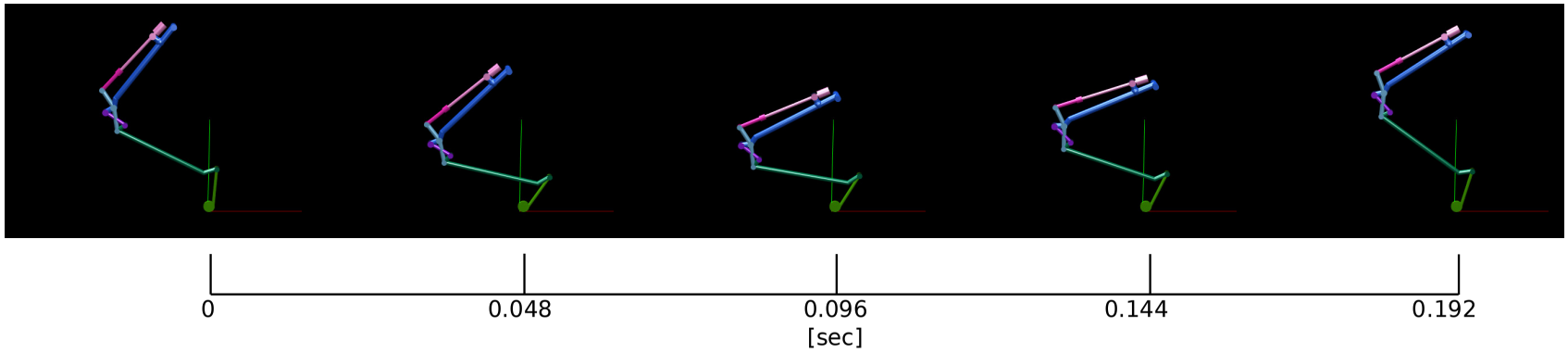

Figure A.11: Best design: Skippy stance phase simulation from $[2.5,-4.5]$ to $[0,4.5]$ m/s.

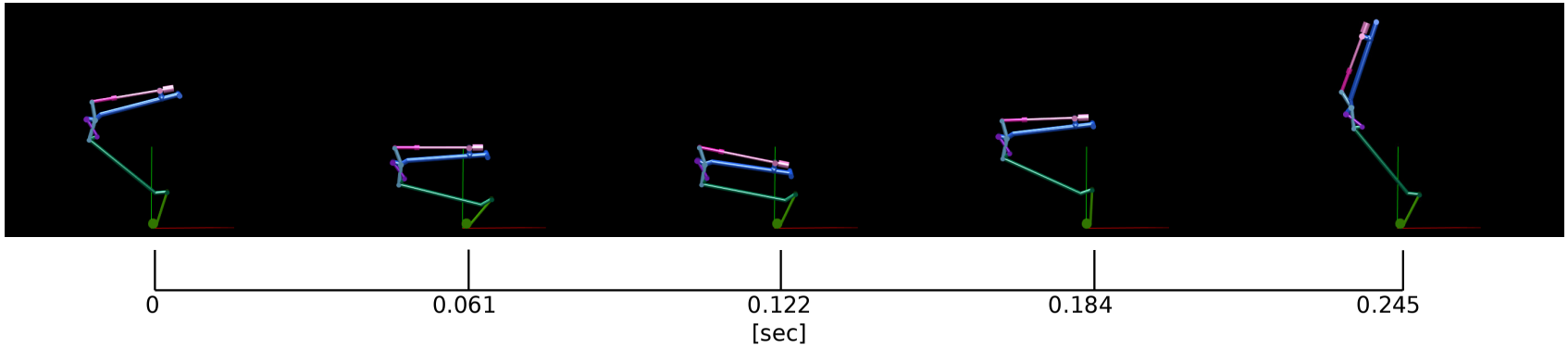

Figure A.12: Best design: Skippy stance phase simulation from $[0,-6.3]$ to $[0,6.3]$ m/s and $3.5\,\mathrm{kgm^2s^{-1}}$.