Università di Pisa

Dipartimento di Ingegneria Civile e Industriale

Corso di Laurea Magistrale in Ingegneria Meccanica

Titolo della Tesi

Direct Trajectory Optimization of Robotic Mechanical Systems with Unscheduled Contact Sequences

Candidato

Tobia Marcucci

Relatori:

Prof. Marco Gabiccini

Prof.ssa Lucia Pallottino

Ing. Hamal Marino

Data di Laurea 30/09/2015

Anno Accademico 2014/2015

Acknowledgements

Anzitutto vorrei sinceramente ringraziare il Professor Marco Gabiccini per il tempo dedicatomi, per la pazienza, per la grande passione con cui mi ha seguito e per tutto quello che mi ha insegnato in questi pochi mesi. Vorrei inoltre esprimere la mia gratitudine nei confronti della Professoressa Lucia Pallottino che, nonostante questo lavoro abbia preso una direzione diversa da quella preventivata, mi ha dedicato molte attenzioni e dato utili consigli. Ringrazio Hamal Marino per la grande disponibilità e per avvermi introdotto ad un settore che mi ha molto appassionato e che spero di poter approfondire nei miei futuri studi. Infine un grazie al Professor Alessio Artoni per l'interesse mostrato, per le osservazioni molto costruttive e per tutto l'aiuto che mi ha fornito.

Abstract

This study evaluates the performances of numerical optimization methods as tools to identify optimal dynamic motions for robotic mechanical systems interacting with the environment through unscheduled contact sequences. Specifically, the attention is focused on a schematic two-dimensional humanoid model, represented as a fixed-base five-DoF articulated serial chain of rigid bodies, in the tasks of getting up (sitting down) from (to) supine and prone positions, opportunistically making and breaking contacts with the ground through hands, elbows, hips, knees, and feet. The different alternatives in the problem transcription, which lead to different constraint equations in the nonlinear program, are determined by: a direct approach to the dynamic problem or a split of the planning into two consecutive optimizations of rising complexity; an embedded introduction of contact force constitutive models in the equations of motion or an augmented definition of the problem, where contact forces are included among the free optimization variables; and a prevention of undesired contacts by the introduction of all the contact forces (even the ones constantly equal to zero) or by the imposition of geometric constraints. Shapes of the emergent behaviors and convergence process sensitivity are evaluated with respect to cost function weights and contact model parameters respectively.

Contents

Chapter 1

Introduction

While successful approaches to dynamic legged locomotion and whole-body motion planning exist, strategies which are general and versatile enough to cope with the variety of situations where inner parts of the body can come into contact, to accomplish a task, with the surrounding environment in a priori unscheduled contact sequences, have appeared in the robotics context only very recently.

Interestingly enough, most of the approaches that proved to be successful in this context are *optimization-based* methods and are often denoted as *direct trajectory optimization* or *direct transcription methods*. The approach is well established in the numerical optimal control community [1, 2], with the most efficient variants being the *direct multiple shooting* [3] and the *direct collocation* [4], where parameterized functions with local support are utilized for both states and inputs.

These methods have been successfully applied in the context of locomotion planning for two-dimensional legged robots and two-dimensional grasp synthesis in [5], where a velocity-based time-stepping scheme is adopted along with a *Linear Complementarity Problem* (LCP) formulation of contact events. A similar approach has been adopted in [6] to synthesize optimal gaits for a planar two-legged robots with series of elastic actuators. Here, the introduction of an higher-order integration scheme for states that are continuous through collisions is the main novelty. In the context of autonomous manipulation, in [7] a framework is devised, based on two different contact models, that allows to synthesize both dexterous and environment-exploiting behaviors in a unified way with encouraging performances.

Other contributions [8] have been directed towards rendering such approaches amenable to whole-body motion planning of three-dimensional robot models by considering the full robot kinematics and condensing the dynamics to the robot's centroidal linear and angular dynamics with impressive outcomes. Along this line, it is worth mentioning the contact-invariant approach, originally proposed in [9], to discover complex behaviors for humanoid figures, and extended to the context of manipulation in [10]. Here, acceptable assumptions (for example massless limbs) to make the problem simpler, and some witty relaxations of the complementarity conditions to allow smooth gradient calculations for contact forces, are paired to a massive use of inverse dynamics, which may lead to inconsistent results for underactuated and/or defective systems. The previously described trajectory optimization method has been recently employed to gradually train a neural network by following an *Alternating Direction Method of Multipliers* (ADMM) strategy with interesting results [11].

To the best of our knowledge, [12] is the only work we found where a quantitative assessment of the performances of direct transcription methods in the context of robotic systems is performed. The investigation includes effects on computational time, quality of the solutions, and sensitivity to open parameters. The feasibility of applying direct transcription methods for online motion planning on a real ballbot robot is included in the analysis.

In this study, we present an extensive investigation of the performances of direct transcription methods, with the collocation scheme, for dynamic motion planning of a humanoid robot model. This is schematically represented as a fixed-base five-DoF articulated serial chain of rigid bodies in two dimensions. The goal consists of performing various whole-body dynamic tasks (rising from the ground from the supine or prone positions or laying down from the standing position) opportunistically making and breaking contacts with the ground through feet, knees, elbows, hands, and even hips if necessary. We could not find in the literature any assessment studies for systems of similar complexity, and with unspecified contact sequences, to compare to. We believe that the unscheduled contact sequence along with the articulated-body system are the key ingredients that render the analysis presented in this paper as complex as interesting.

1.1 Different Approaches to the Motion Planning Problem

As pointed out this study is focused on the application of direct transcription methods but many other techniques to approach motion planning problems has been proposed in recent years. In this section we overview probably the two best known families of optimal control algorithms for nonlinear systems: the first group we introduce is the one that include direct transcription and it is generally referred to as *numerical optimization methods*, then *sampling-based methods* are illustrated.

1.1.1 Numerical Optimization Methods

Numerical optimization methods concern with the solution of *Continuous Time Optimal Control Problems* (CTOCP). Given a description of the dynamic system in terms of *Ordinary Differential Equations* (ODE), this problem can be stated as

$$\begin{array}{lll} \underset{x,u}{\text{minimize}} & \displaystyle\int_0^T L(x(t),u(t))dt + E(x(T)) & \text{(objective function)} \\ \text{subject to} & x(0) - x_0 = 0 & \text{(fixed initial value)} \\ & \dot{x}(t) - f(x(t),u(t)) = 0,\ t \in [0,T] & \text{(ODE model)} \\ & h(x(t),u(t)) \leq 0,\ t \in [0,T] & \text{(path constraints)} \\ & r(x(T)) \leq 0 & \text{(terminal constraints)} \end{array}, \tag{1.1}$$

where the integral cost contribution $L(x(t), u(t))$ is called *Lagrangian term* of the objective function and the contribution $E(x(T))$ is called *Mayer term* of the objective function, whereas the sum of both is generally referred to as a *Bolza objective*.

More generic formulations of Problem 1.1 can be stated, for example: every function can depend from time explicitly, some of the system parameters can be included in the objective function (even the planning time T could be considered as a variable of optimization) or *Differential Algebraic Equations* (DAE) models can be used. On the other hand, note that considering only inequality path constraints is not a restriction (an equality relation can be imposed using two inequalities).

Generally speaking, we can distinguish three families of methods to address Problem 1.1: *state-space methods*, *indirect methods* and *direct methods*.

- State-space methods are based on the *principle of optimality*: since objectives are integral functions (therefore the additive property holds), any subarc of an optimal trajectory is optimal itself. Taking advantage of this principle is possible to split Problem 1.1 in a set of short horizon problems, which can be first computed recursively backwards and then brought back together to obtain the optimal solution. This approach is called *Dynamic Programming* (DP) in discrete time cases, meanwhile it leads to *Hamilton-Jacobi-Bellman equations* (HJB) for continuous time systems. Probably the greatest advantage of state-space methods is their ability to find global minima, on the other hand the application of these methods is restricted to small scale systems.

- Indirect methods (also known as "first optimize, then discretize") start from the HJB equations and, exploiting the *Pontryagin's minimum principle*, build up a boundary value problem. This system of equations must be solved numerically and it contains strongly nonlinear and unstable ODEs. In comparison with DP, it is possible to manage systems of larger scale, but the obtained solution is a local optimum.

- Direct methods (also known as “first discretize, then optimize”) transform the continuous time problem into a finite dimensional *NonLinear Program* (NLP) that can be solved using high efficient numerical algorithms. All direct methods have in common the phase of discretization of the control functions, differences lie in the way that the evolutions of the states are estimated. In this case the principal advantage is the robustness and the relatively easy application: NLP solvers deal with inequalities and active set changes with great effectiveness. Unfortunately the solution is a local minimum.

Nowadays direct methods are the most commonly used for real applications and probably the most successful technique belonging to this family are direct collocation and direct multiple shooting. For this reason, during this study, we decided to focus our attentions on these two methods; in Chapter 2 we will precisely illustrate and analyze their structures and principal properties.

1.1.2 Sampling-Based Methods

Sampling-based methods approach the optimal control problem from a complete different prospective. In this case the solution of the motion planning problem is reached by sampling the configuration space ($\mathcal{C}$) and trying to connect samples with trajectory pieces that do not collide with obstacles (belong to $\mathcal{C}_{free}$). While the number of samples increases, the connections between each milestone (sample that belong to $\mathcal{C}_{free}$) create a so called road map that collects all the safe trajectories to connect two states. When the road map reaches the goal configuration, a solution to the control problem is found (even if it certainly is not an optimal one). Going on probing $\mathcal{C}$, a better solution might be found but only few methods provide this guarantee; as we will illustrate, some algorithms are demonstrated to converge to the absolute optimal solution as the number of samples tends to infinity (in this case their are said to be *asymptotic optimal*).

As pointed out, optimality is a characteristic that an efficient motion planning algorithm must have, but this is only one of many. Another fundamental property is the *completeness*, an algorithm is said to be complete if it terminates in a finite time returning: a right solution if one exists, failure otherwise. First complete planning algorithms were developed in late 70s but, at the beginning, their complexity made them unsuitable for practical applications. Some years later, with the introduction of cell decomposition methods and potential fields, concrete approaches were proposed. Even if these methods demonstrated to perform efficiently within reasonable time bounds, their need to explicitly express obstacles in $\mathcal{C}$ resulted in an excessive computational complexity. Avoiding an explicit representation of the obstacles is one of the main idea behind sampling-based methods; these algorithms implement a “black box” collision detection module that uses models such as semi-algebraic sets, three-dimensional triangles, nonconvex polyhedra and so on. This gives the possibility to develop algorithms that are totally independent of the particular obstacle models. Speaking about completeness, some sampling-based method are *probabilistic complete*, in the sense that the probability of the algorithm to find a feasible solution tends to one as the samples approach infinity.

The best known sampling-base algorithms are the *Probabilistic Road Maps* (PRM) and the *Rapid-exploring Random Trees* (RRT). These two methods approach similarly the sampling phase but they differ in the way they connect samples. PRM is said to be a multiple-query method, in the sense that it develops a road map that can be used in case of different goal configurations, this fact obviously reflects on the computational cost of the algorithm. PRMs have been reported to perform well in high dimensional spaces and they also guarantee probabilistic completeness, on the other hand it is important to note that most online planning problems do not require multiple queries. For all these reasons single-query incremental algorithms prevailed in recent years. RRT is likely the most popular of these methods, some of its features are the effectiveness in differential constraints handling and the probabilistic completeness.

Speaking about optimality, none of the methods presented so far guarantee this characteristic. Only in 2011 a complete and rigorous analysis of these, and many others, properties was presented [13]. In this study not only the non-optimality of PRM and RRT (and many others algorithms) is proved, but new algorithms denominated PRM* and RRT* are presented. These new highly efficient methods are demonstrated to be probabilistic complete and asymptotical optimal (in addition to their reduced time and space complexity).

Even though the concepts presented in this section are only the very bases of sampling-based motion planning, it is clear how this family of methods allow a very pragmatic approach to these topics; this fact is probably one of the main cause of the recent increase of interest in them. On the other hand we underline their great capability to solve planning problems in high dimensional spaces in relatively short times. A complete introduction to sampling-based methods can be found in [14].

1.2 Case Study and Original Aims

In the recent years great progresses has been made in the field of motion planning and a variety of methods of different nature has been proposed but, at the moment, none of them has really prevailed over the others. Different branches of mathematics has been examined and lots of efforts were made to exploit well known control theories to obtain revolutionary approaches to motion planning problems.

Inspired by all these works, we decided to analyze the possibility of mixing the methods presented above. The original title of this thesis was "An Hybrid Approach to Dynamic Planning of Robotic Mechanical Systems with Intermittent Contacts" where, indeed, "Hybrid" represented our intention to combine direct transcription with sampling-based methods, in order to bring out the best in each one.

Numerical optimization methods exploit Newton-type algorithms, that guarantee a very fast convergence to the optimum. Unfortunately they have some weak points: neither completeness nor optimality are guaranteed and the computed solutions are strongly dependent on the initial guess (even if this dependance can sometimes play the useful role to guide the results to assume specific characteristics). On the contrary some sampling-based methods are proved to be probabilistic complete, asymptotic optimal and they obviously don't need an initial guess. Considering that, an implementation of an hybrid method seems a good idea. For example, we can think to a parallel use of the two methods (while sampling-based algorithm converges to the absolute optimum, a numerical optimization of the temporary trajectories can be made) or to initialize a Newton-type search with a set of feasible trajectory pieces. Unfortunately, because of some unforseen difficulties, our study has been limited to the application of the numerical optimization methods.

In order to test our ideas we decided to study a specific case which at the time appeared simpler than it really is. Inspired by the great amount of studies made on humanoids, we chose to analyze a simplified two-dimensional robot (illustrated in Figure 1.1 [1]) composed by a chain of links connected exclusively by revolute joints. At first we focused our attentions on the planning of a rising motion; in particular the first task we analyzed starts from a supine position, with the hands and the hips in contact with the ground, and finishes in a standing configuration (exactly the one represented in Figure 1.1). As we will see in Chapter 4, after a fine tune of this case we decided to move on and consider a wider range of tasks.

A two-dimensional model of the robot implies that every movement has to be symmetrical about the plane of representation, such that left and right limbs can't move

[1]humanoid pictures are developed in Mathematica environment [18].

independently. This assumption is extremely limiting in order to study any kind of realistic motion but it drastically reduce the complexity of the problem and, as we will discuss in this relation, complexity is actually the greatest limit for these kinds of motion planning methods. On the other hand, considering the fact that we are interested in the study of rising/sitting movements, this hypothesis appears to be more appropriate.

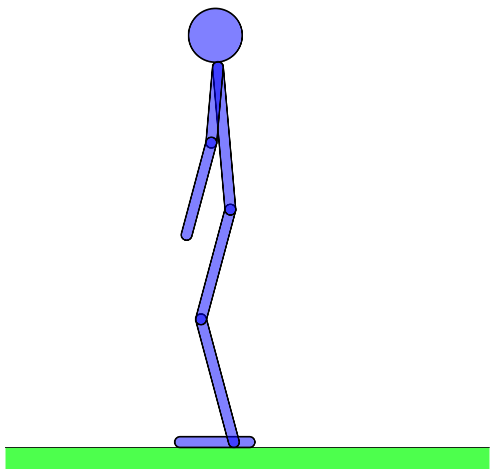


Figure 1.1: Simplified two-dimensional humanoid.

At first we tried to approach the problem modelling the humanoid with floating feet; soon we realized that this model was too complex and some simplifications were needed. Analyzing some intermediate results we established to reduce the degree of freedom from 8 to 5, for this reason feet were removed and their place was taken by a fixed hinge (equipped wiht a set of constraints that guarantee the static equilibrium of the removed feet). With some other simplifications (illustrated in the following chapters) we finally obtain a manageable problem.

Chapter 2

Direct Methods to the Solution of Optimal Control Problems

Working on the example introduced in Section 1.2 two direct methods of numerical optimization has been considered: direct collocation and direct multiple shooting. As said, all direct methods have in common a finite dimensional parametrization of the input functions and the main difference lies in the way that state trajectories are computed. In this chapter we introduce and compare these two methods.

The final result of these direct method is an NLP (approximation of the CTOCP) that has the general form

$$\begin{aligned} \underset{v}{\text{minimize}} \quad & \phi(v) && \text{(objective function)} \\ \text{subject to} \quad & \chi(v) = 0 && \text{(equality constraints)} \quad , \\ & \psi(v) \leq 0 && \text{(inequality constraints)} \end{aligned} \tag{2.1}$$

where v is the vector that collects all the optimization variables. In Section A.2 we present the technique we use to solve these problems: *Interior-Point Method* (IPM).

2.1 Direct Collocation

Direct collocation is probably the most straightforward direct method. We start discretizing the time axis on a fine grid $[t_0, t_1, ..., t_N]$, control functions are assumed to be constant during each period ($u(t) = w_i$ for $t \in [t_i, t_{i+1}]$ and $i \in \{0, 1, ..., N-1\}$) and states are approximated with their node values $s_i \approx x(t_i)$ for $i \in \{0, 1, ..., N\}$. We replace the ODE model in the CTOCP (Problem 1.1) with a finite set of equality constraint equations

$$c_i(s_i, s_{i+1}, w_i) = 0, \ i \in \{0, 1, ..., N-1\}.$$

For example, in case of a system expressed in state space form, an efficient way to write these constraints is: $c_i(s_i, s_{i+1}, w_i) = \frac{s_{i+1} - s_i}{\Delta t} - f(\frac{s_{i+1} + s_i}{2}, w_i)$; where $\Delta t = t_{i+1} - t_i$.

The next step is the approximation of the integral terms of the objective function

$$\int_{t_i}^{t_{i+1}} L(x(t), u(t)) \approx l_i(s_i, s_{i+1}, w_i),\ i \in \{0, 1, ..., N-1\}.$$

where, for example, $l_i(s_i, s_{i+1}, w_i) = L(\frac{s_{i+1}+s_i}{2}, w_i)\Delta t$.

Substituting these relations in the original Problem 1.1 we obtain a NLP that has the form

$$\begin{array}{lll}
\underset{s,w}{\text{minimize}} & \sum_{i=0}^{N-1} l_i(s_i, s_{i+1}, w_i) + E(s_N) & \text{(objective function)} \\
\text{subject to} & s_0 - x_0 = 0 & \text{(fixed initial value)} \\
& c_i(s_i, s_{i+1}, w_i) = 0 & \text{(discretized ODE model)} \\
& h(s_i, w_i) \leq 0 & \text{(discretized path constraints)} \\
& r(s_N) \leq 0 & \text{(terminal constraints)}
\end{array}, \tag{2.2}$$

where $i \in \{0, 1, ..., N-1\}$ whereas $s = [s_0, s_1, ..., s_N]$ and $w = [w_0, w_1, ..., w_{N-1}]$ represent the arrays that collect the values of the controls and the states at each step.

Substituting $v = [s_0, w_0, ..., w_{N-1}, s_N]$ we reduce Problem 2.2 to the standard structure of Problem 2.1.

2.2 Direct Multiple Shooting

Unlike collocation method, using multiple shooting we have to perform a coarse discretization of time axis and, for each interval, numerically solve the ODE (model of the system)

$$\begin{aligned}
&\dot{x}_i(t; s_i, w_i) = f(x_i(t; s_i, w_i), w_i) \\
\text{subject to}\quad &x_i(t_i; s_i, w_i) = s_i
\end{aligned}$$

for $t \in [t_i, t_{i+1}]$ and $i \in \{0, 1, ..., N-1\}$.

Solving latter ODEs (starting with artificial initial conditions s_i) we obtain a set of trajectory pieces $x_i(t; s_i, w_i)$, whose continuity guarantee the respect of the discretized dynamic model.

For each trajectory piece, we calculate the Lagrange term of the cost function

$$l_i(s_i, w_i) = \int_{t_i}^{t_{i+1}} L(x_i(t; s_i, w_i), w_i)dt.$$

At this point Problem 1.1 assumes the form

$$\begin{array}{lll} \underset{s,w}{\text{minimize}} & \displaystyle\sum_{i=0}^{N-1} l_i(s_i, w_i) + E(s_N) & \text{(objective function)} \\ \text{subject to} & s_0 - x_0 = 0 & \text{(fixed initial value)} \\ & s_{i+1} - x_i(t_{i+1}; s_i, w_i) = 0 & \text{(continuity constraints)} \\ & h(s_i, w_i) \leq 0 & \text{(discretized path constraints)} \\ & r(s_N) \leq 0 & \text{(terminal constraints)} \end{array} , \qquad (2.3)$$

where $i \in \{0, 1, ..., N-1\}$.

Once again, the substitution $v = [s_0, w_0, ..., w_{N-1}, s_N]$ reduces Problem 2.3 to the standard structure of Problem 2.1

The last operation we illustrate is the numerical solution of the ODEs. A great variety of integrators can be used, in our case we decided to utilize the *Explicit Runge-Kutta* 4^{th} *order method* (ERK4), that can be summarized as follows.

Each one of the previous N intervals is now divided into n steps. Using the index $j \in \{0, 1, ..., n-1\}$ to number these additional periods and denoting by $\delta t = \frac{t_{i+1} - t_i}{n}$ their duration, we approximate the generic j^{th} value of the state as

$$\begin{aligned} & x_{i,j+1} = x_{i,j} + \frac{\delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4), \\ \text{where} \quad & k_1 = f(x_{i,j}, w_i), \\ & k_2 = f(x_{i,j} + k_1 \frac{\delta t}{2}, w_i), \\ & k_3 = f(x_{i,j} + k_2 \frac{\delta t}{2}, w_i), \\ & k_4 = f(x_{i,j} + k_3 \delta t, w_i). \end{aligned}$$

Starting with the initial value $x_{i,0} = s_i$, each trajectory $x_i(t; s_i, w_i)$ is computed in the period $[t_i, t_{i+1}]$.

2.3 Performance Comparison

In this section we briefly report the principal properties of the methods presented above, pointing out the characteristics that brought us to choose direct collocation instead of direct multiple shooting.

Starting from the technicality, the first feature to examine is the sparsity of the NLPs. A NLP is said to be sparse if the Jacobian matrices $\nabla_v \chi(v)^T$ and $\nabla_v \psi(v)^T$ contain many zero elements; taking advantage of this peculiar structure, solvers can increase their efficiency and drastically reduce the computation time. Collocation methods lead to a large scale but very sparse NLP, on the other hand multiple shooting produces a more compact NLP not as sparse as the collocation one. Both methods treat unstable systems efficiently and handle path and terminal constraints very robustly. Speaking of inequality constraints for multiple shooting method, it is important to note that, in order to check inequalities on a finer grid than the one introduced to discretize the inputs, it is necessary to have continuous outputs from the integrator (aspect that considerably complicate the implementation of the problem).

An important aspect, that can seriously complicate the application of a multiple shooting method, is that the great majority of the ODE solvers available in common use frameworks (like ERK4) need as input an ODE in state space form. On the other hand the use of a collocation method isn't affected by this limitation, in fact in this case the dynamic constraints are introduced as a set of algebraic equation that the discretized optimization variables have to respect. Furthermore this is not only a practical advantage but it also leads to an increase of performances: for example we can think to have the ODE model of a mechanical system in the *Lagrange standard form*

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} = Q(q,\dot{q},u),$$

where q represents the configuration vector. This system of equations can easily be converted in a state space form, but the inversion a large symbolic matrix generates a huge matrix (that has to be multiplied to the remaining vectors). This inefficiency can be avoided by discretizing the equations of motion (at the generic i^{th} step for $i \in \{0, 1, ..., N-1\}$) (for example) in the very simple way

$$B(q_{i+1})\frac{\dot{q}_{i+1} - \dot{q}_i}{\Delta t} + C(q_{i+1}, \dot{q}_{i+1})\dot{q}_{i+1} = Q(q_{i+1}, \dot{q}_{i+1}, u_i)$$

then adding a relation between configurations and velocities, like the one obtained with the *backward Euler method*

$$\dot{q}_{i+1} = \frac{q_{i+1} - q_i}{\Delta t}.$$

To conclude this parenthesis we underline the fact that some methods to write equations of motion (for example the *Newton-Euler recursive method*) lead to a non-factorized results that, for large scale systems, can be very complex to put in state space form.

As revealed earlier, we ended up choosing the collocation method for the example of the two-dimensional humanoid. This approach has demonstrated to be approximately five times faster than multiple shooting (with a comparable precision).

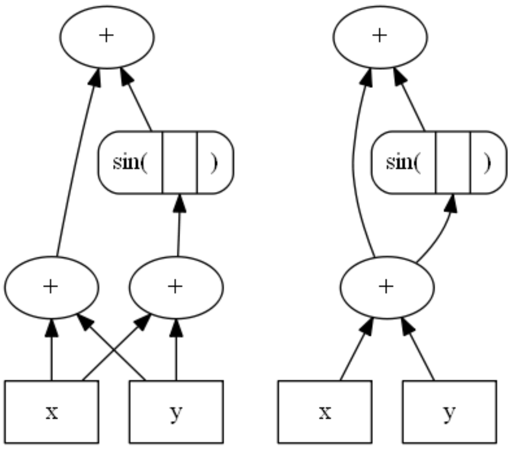
Chapter 3

Dynamic Analysis of a Simplified Two-Dimensional Humanoid

In this chapter we derive the equations of motion for the robotic system presented in Section 1.2. We chose to approach the problem using the Lagrange standard form for a series of reasons. The first advantage lies in the fact that the obtained equations are factorized or, more in detail, the inertia matrix isn't multiplied by the vector of accelerations; this allows us to quickly switch the algorithm of optimization (as pointed out in Chapter 2 the use of multiple shooting methods is very simplified if the ODE model can be put in state space form). Another advantage of having factorized equations is the possibility to rapidly eliminate terms that, in specific cases, are considered negligible (in Chapter 4 we will frequently adopt the procedure of removing dynamic terms from the equations of motion). In addition to the derivation of dynamic equations, the necessary constraints to correctly represent the static actions applied to the removed feet are illustrated. Moreover two brief parenthesis are made: first the influence of the data structures on the efficiency of the optimization algorithms is discusses, at the end an alternative idea to impose dynamic constraints in the optimization problem is presented.

3.1 Graph Data Structures

Before starting to talk about kinematics and dynamics it is fundamental to face the data structure topic. We developed our codes using CasADi [16]: a symbolic framework for algorithmic differentiation and numeric optimization. One of the main features of CasADi is the use of graph structures (as opposed to trees) to organize data. Graphs avoid to occupy memory with unnecessary copies of identical expressions, this is done by creating connections between different branches of the graph in case that the same intermediate result is needed (as input) in several further operations. This organization of the memory leads to a great economy in the number of computations and gives the possibility to approach very large scale systems. In order to clarify this concept, it's useful to analyze a trivial example. Let us consider the symbolic expression $x + y + \sin(x + y)$, whose graph is the one in Figure 3.1a [1]. As we can see it is composed by 6 nodes and 4 elementary operations are needed, but it's clear that an unnecessary copy of the term $x + y$ is generated. With the goal to reduce the number of nodes, we can define a new symbolic variable $z = x + y$ and write the original expression as $z + \sin(z)$, the graph we obtain is the one illustrated in Figure 3.1b. As we can see the redundant node is eliminated and the variable z is computed only once.



(a) x+y+sin(x+y). (b) $z + \sin(z)$.

Figure 3.1: Different graphs related to the same expression.

This might seem a too trivial example but understanding this logic is crucial: during the development of our codes we ran into a series of problems caused by the excessive length of the dynamic equations we derived and lots of attempts were made in order to simplify and compact them. The most time-waster try we performed was indeed related to a lack of knowledge of these topics: we decided to export intermediate results from CasADi, import them in Mathematica, exploit Mathematica's simplification routines and then bring the equations back to CasADi. Even if the obtained expressions were approximately one tenth length than the originals, graphs generated by reading (all at once) these (not as huge as before, but still large) inputs were extremely disorganized. Only after some time we realized that the simplified equations generated a greater number of nodes than the original expressions.

These considerations made us realize that a more efficient and pragmatic way to write equations was needed. We have not found an overall valid solution to this problem, and for sure our future attentions will be (among other things) directed to that way. Temporarily

[1]graph pictures are obtained using Graphviz's Dot, with Python interface pydot.

our life-saver is to analyze the structure of every geometric and dynamic function we derive and (whenever possible) try to express it in a closed form; in these cases a tailored "graph-compressing" implementation can be made. In order to have an rough idea, our first approach to the problem (with 8 DoF and floating feet) leads to dynamic equations (in state space form) with 1.4 millions nodes, while the final (with a fixed hinge in place of the ankles and some expressions introduced in a closed form) has 8.2 thousands of nodes: almost a factor of 10^3!

3.2 Dimensions and Parameters

We start introducing the principal lengths and parameters of the humanoid illustrated in Figure 3.2 [2]. The values (listed in Table 3.1) are elaborations of the segment parameters presented in [15] (an adult man with a body mass of 73.0 kg and 1.741 m height is considered). Despite the fact that humanoid robots have quite different mass distributions, we consider this approximation admissible in order to capture principal dynamic behaviours.

Link number	**a** (m)	**c** (m)	**m** (kg)	**i** ($\mathrm{kg} \cdot \mathrm{m}^2$)
1	0.473	0.208	6.32	0.0855
2	0.422	0.173	20.7	0.399
3	0.532	0.179	36.8	1.92
4	0.282	0.119	3.96	0.0227
5	0.355	0.174	3.26	0.0434

Table 3.1: Humanoid lengths and parameters; respectively: lengths, center of mass positions, mass and moment of inertia.

Referring to Figure 3.2, the humanoid head is considered to be fixed to the trunk (Link 3) and its inertia is taken into account in the computation of Table 3.1 parameters, in the same way hands parameters are included in the forearms (link 5) data.

3.3 Geometric Functions

In this section we present the essential geometric functions required to obtain dynamic equations. We start analyzing homogeneous transformation matrices: since our problem is two-dimensional and only revolute joints are present, these matrices can easily be

[2] schematic images of the humanoid and its body parts are developed using Ti*k*z TEXpackage.

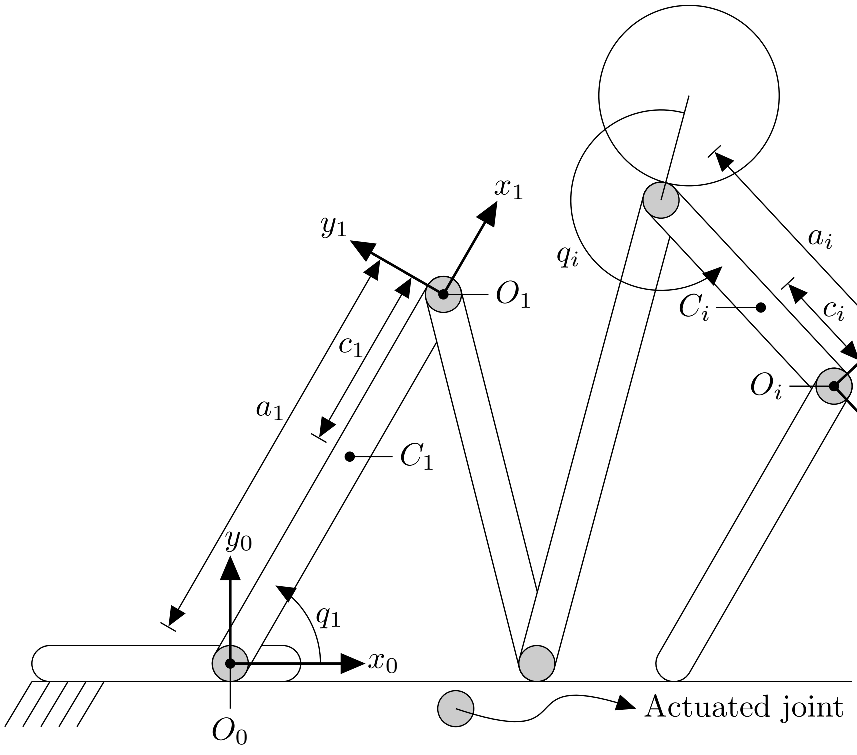


Figure 3.2: Humanoid geometry.

expressed in a closed form. Calling the configuration vector $q = [q_1, q_2, ..., q_n]^T$, we obtain the following expressions:

$$T_{i-1,i} = \begin{bmatrix} \cos q_i & -\sin q_i & a_i \cos q_i \\ \sin q_i & \cos q_i & a_i \cos q_i \\ 0 & 0 & 1 \end{bmatrix},$$

$$T_{0,i} = \prod_{j=1}^{i} T_{j-1,j} = \begin{bmatrix} \cos(\sum\limits_{j=1}^{i} q_j) & -\sin(\sum\limits_{j=1}^{i} q_j) & \sum\limits_{k=1}^{i} a_k \cos(\sum\limits_{j=1}^{k} q_j) \\ \sin(\sum\limits_{j=1}^{i} q_j) & \cos(\sum\limits_{j=1}^{i} q_j) & \sum\limits_{k=1}^{i} a_k \sin(\sum\limits_{j=1}^{k} q_j) \\ 0 & 0 & 1 \end{bmatrix},$$

where, for example, $T_{i-1,i}$ represents the transformation between the frame fixed to the link i^{th} and the one fixed to the link $i-1^{th}$ (Figure 3.2).

In order to show what is the form of a more useful graph than the ones illustrated before, in Figure 3.3 we present the graph related to the the term $[1, 3]$ of $T_{0,i}$ (that represent the absolute x_0 coordinate of the point O_5). As we can see this graph isn't as compressed as possible, however its representation can be considered acceptable: further reductions of its dimensions would be as challenging as dispersive.

Starting from the expression of $T_{1-i,i}$ and $T_{0,i}$, we can define the geometric Jacobian as the linear transformation that maps the time derivative of the configuration vector q in

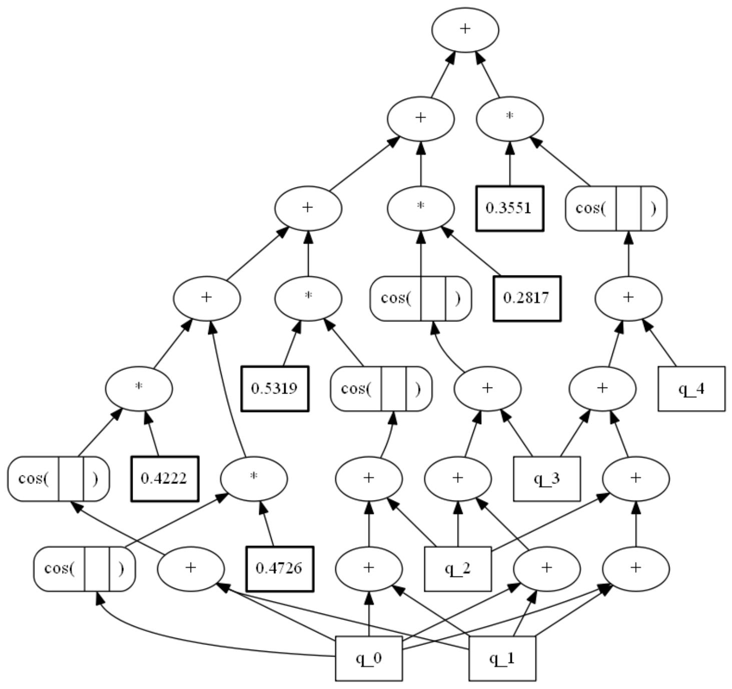


Figure 3.3: Graph representing the computational process related to the absolute x_0 coordinate of the point O_5.

the linear and angular velocity of a specific point P_i (that is considered fixed with respect to the i^{th} link). Let us call twist of the point P_i the vector where linear and angular velocity (respectively of the point and the relative body) are vertically stacked; let us also indicate it as $\xi_{P_i} = [v_{P_i}^T \quad \omega_i^T]^T$. The geometric Jacobian has the following role:

$$\xi_{P_i} = J_{P_i}(q)\dot{q} = \begin{bmatrix} J_{v_{P_i}}(q) \\ J_{\omega_i}(q) \end{bmatrix} \dot{q}$$

where $J_{v_{P_i}}(q)$ is called position Jacobian and $J_{\omega_i}(q)$ is called orientation Jacobian.

The generic column j of $J_{P_i}(q)$, if the j^{th} joint is a revolute one, has the following form.

$$J_{P_i}(q)[:,j] = \begin{cases} [(k_{j-1} \times (O_{j-1}P_i))^T \quad k_{j-1}^T]^T & \text{if } j \leq i \\ [0^T \quad 0^T]^T & \text{if } j > i \end{cases}$$

where k_{j-1} represents the unitary vector associated to the z_{j-1} axis.

The next step is to derive the expressions of the geometric Jacobians of each point O_i and C_i (see Figure 3.2) expressed in the absolute frame of reference. Their expression can be derived simply substituting O_i (or C_i) to P_i in the latter equation and expressing each vector in the absolute frame. It is obvious that, since our problem is two-dimensional, each orientation Jacobian is composed by zeros except for the third row that has the first i elements equal to 1. The terms of the position Jacobians are less obvious: by inspection, it's possible to see that the j^{th} column (for $j \leq i$) of $J_{v_{O_i}}(q)$ has the form (focusing on the in-plane velocities)

$$J_{v_{O_i}}(q)[:,j] = \begin{bmatrix} -\sum\limits_{k=j}^{i} a_k \sin(\sum\limits_{l=1}^{k} q_l) \\ \sum\limits_{k=j}^{i} a_k \cos(\sum\limits_{l=1}^{k} q_l) \end{bmatrix}.$$

A completely equivalent exression can be found for $J_{v_{C_i}}(q)[:,j]$ with the "trick" of substituting a_i with $a_i - c_i$. We will refer to this array of lengths as $a^i \in \mathbb{R}^i$ (where the superscript can be different from i), that is:

$$a_k^i = \begin{cases} a_k & \text{if } k < i \\ a_k - c_k & \text{if } k = i \end{cases}.$$

The last geometric entity we introduce will be necessary to compute reaction forces applied in the fixed hinge: the geometric Jacobian of the whole system center of mass ($J_{v_C}(q)$, we will omit subscripts to refer to the whole system properties). Starting from the definition of the center of mass position (expressed in the absolute frame)

$$C = \frac{1}{m}\sum_{i=1}^{n} m_i C_i$$

we can obtain its velocity by differentiation

$$v_C = \dot{C} = \frac{1}{m}\sum_{i=1}^{n} m_i v_{C_i} = \left(\frac{1}{m}\sum_{i=1}^{n} m_i J_{v_{C_i}}(q)\right)\dot{q} = J_{v_C}(q)\dot{q}.$$

Exploiting this matrix, the acceleration of the center of mass (necessary to compute inertial forces that are equilibrated by the fixed hinge reaction forces) assumes the form

$$a_C = \dot{v}_C = \frac{\partial(J_{v_C}(q)\dot{q})}{\partial q}\dot{q} + J_{v_C}(q)\ddot{q}. \tag{3.1}$$

3.4 Contact Force Model

A fundamental aspect of planning with intermittent contacts is the model of contact force which is exploited: since contact forces are non-smooth by nature, they can't be directly handled by Newton-type algorithms. We can think of two main alternatives to deal with this discontinuity: approximate the contact forces with differentiable functions that tend to the real shape as some parameters approach specific values or define these forces as optimization variables and, thanks to a set of LCP complementarity conditions, pass the burden of the non-differentiability to the optimization algorithm. Because of the simplicity of the first strategy, in this study we decided to approach the problem that way; in this section we present the functions we developed to model normal and tangential contact force. In this simple case each link is considered to be a line (in the sense that it is characterized only by its length) and for this reason, considering a flat ground, contact forces can be applied only to the extremity of each link.

Starting from normal contact force, we tried several functions to approximate the rigid contact, such as hyperbolas or exponentials but we will present only the model we used most. With the goal to make a common linear elastic unilateral contact differentiable, we developed the function (whose representation is compared to a linear function in Figure 3.4 [3])

$$F_n = \frac{F_0}{\log(2)} \log\left(1 + 2^{-\frac{\kappa}{F_0}y}\right),$$

where y is the normal gap, F_0 is the value of force for $y = 0$, $-\kappa$ is the stiffness of the contact as $y \to -\infty$, and $\log(2)$ is a constant whose presence is necessary as a consequence of the use of F_0 and κ.

Coefficients make it look more complicated than it really is: a linear unilateral contact force has discontinuous derivative ($-\kappa$ if $y \leq 0$, 0 if $y > 0$) that can be approximated by an hyperbolic function ($\kappa\frac{\tanh(y)-1}{2}$) whose primitive looks like $\log(1 + e^{-y})$.

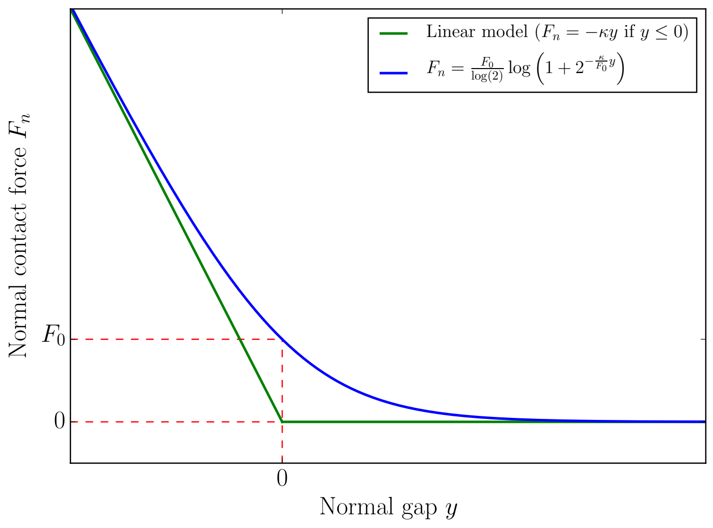


Figure 3.4: Normal contact force as a function of normal gap.

Friction contacts are modelled with an hyperbolic approximation of Coulomb law (illustrated in Figure 3.5):

$$F_t = -\mu F_n \varphi(\dot{x}), \text{ with } \varphi(\dot{x}) = \tanh\left(\frac{\dot{x}}{\dot{x}_{\text{ref}}}\right),$$

where μ is the friction coefficient, $\dot{x}$ is the tangential velocity, and $\dot{x}_{\text{ref}}$ is an opportune value of $\dot{x}$ at which the the tangential force F_t is 76% of its asymptotic value.

[3]graphs are developed using matplotlib Python library.

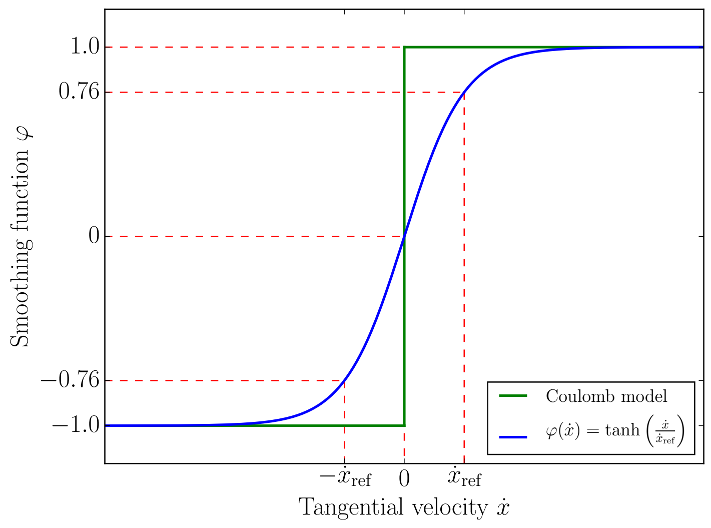


Figure 3.5: Friction coefficient as a function of tangential velocity.

Defined the models of contact forces we are going to use, we have to choose the numerical values of the parameters we introduced. Starting from μ we decided to use a friction coefficient equal to 1, considering that for this kind of movements high values of friction forces are required. The choice of the value to adopt for F_0 must take into account some aspects: the use of smooth contact forces sometimes has the consequence that initial (or goal) positions do not respect constraints imposed to the fixed ankles actions moreover, even the times when these constraints are respected, these models might lead to high values of input torques to maintain these positions. In this sense we have to guarantee a value of F_0 such that the weight of the humanoid (697 N) is balanced with almost zero penetration in the ground and, considering that most of the time the robot has two contacts, we found 400 N to be an effective compromise. Moving on to the definition of κ and $\dot{x}_{\mathrm{ref}}$, we decided to adopt a stiffness equal to $5.0 \cdot 10^4 \ \frac{\mathrm{N}}{\mathrm{m}}$ and a reference to the tangential velocities of $5.0 \cdot 10^{-2} \ \frac{\mathrm{m}}{\mathrm{s}}$. The choice of these two values is in depth validated in Section 5.1 where a detailed analysis of the effects of these two parameters on the convergence process is proposed.

3.5 Dynamic Equations

As said before, several reasons made us choose to write equations of motions in Lagrange standard form

$$B(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = Q(q, \dot{q}, u),$$

where q is the configurations vector, B is the inertia matrix, C is the Coriolis and centrifugal terms matrix, G is the gravitational forces vector, and Q is the vector of nongravitational external forces.

It is proved that the inertia matrix can be expressed as:

$$B(q) = \sum_{i=1}^{n} (m_i J_{v_{C_i}}(q)^T J_{v_{C_i}}(q) + J_{\omega_i}(q)^T \mathcal{I}_i J_{\omega_i}(q))$$

where $\mathcal{I}_i$ is the moment of inertia tensor of the link i (note that, since our problem is two-dimensional, we can avoid the introduction of tensors by using scalar in-plane moments of inertia).

In this specific case $B(q)$ assumes a particular pattern that is described by the following expressions. Splitting the inertia matrix into two parts (the one associated to linear velocities and the one related to angular velocities) and considering the generic $[r, s]$ term, we obtain

$$B[r, s] = M[r, s] + I[r, s] = \sum_{l=1}^{n} (M_l[r, s] + I_l[r, s]),$$

where M_l represents the contribution of each link linear velocity to the global inertia matrix. By inspection it can be seen that M_l has the form

$$M_l[r, s] = \begin{cases} m_l \left(\sum\limits_{i=\max(r,s)}^{l} (a_i^l)^2 + \sum\limits_{\substack{i=r \\ j=s \\ i \neq j}}^{l} a_i^l a_j^l \cos \left(\sum\limits_{k=\min(i,j)+1}^{\max(i,j)} q_k \right) \right) & \text{if } r \leq l \wedge s \leq l \\ 0 & \text{if } r > l \vee s > l \end{cases},$$

whereas the contribution of each link angular velocity to the global inertia matrix is trivially the related moment of inertia

$$I_l[r, s] = \begin{cases} i_l & \text{if } \max(r, s) \leq l \\ 0 & \text{if } \max(r, s) > l \end{cases}.$$

Coriolis and centrifugal terms are obtained using the Christoffel symbols

$$C[i, j] = \sum_{k=1}^{n} \Gamma^i_{jk} \dot{q}_k,$$

where

$$\Gamma^i_{jk} = \frac{1}{2} \left(\frac{\partial B[i, j]}{\partial q_k} + \frac{\partial B[i, k]}{\partial q_j} - \frac{\partial B[j, k]}{\partial q_i} \right).$$

The j^{th} term of the gravitational torques vector can be easily determined

$$G_j = -\sum_{i=1}^{n} m_j J_{v_{C_i}}[:, j]^\top g,$$

where $g = [0 \quad -9.81]^\top \ \frac{\mathrm{m}}{\mathrm{s}^2}$.

Finally the contribution of the remaining external generalized forces can be expressed as

$$Q_j = \sum_{i \in A_{\mathrm{cf}}} J_{v_{O_i}}[:, j]^\top F_{c_{O_i}} + u_j,$$

where $F_{c_{O_i}} = [F_{t_{O_i}} \quad F_{n_{O_i}}]^\top$. Furthermore, in order to reduce the computational burden associated to undesirable contacts, the set of the active ("turned on") contact forces $A_{\mathrm{cf}} \subseteq \{1, \ldots, n\}$ is changed from test to test. For example, considering the motions presented in Section 4.2, the contact force applied to the shoulders of the humanoid will never be taken into account, while the forces applied on the knees, the hips, and the elbows will be considered only in some cases.

3.6 Fixed-Base Assumption

As exposed before, in order to reduce the complexity of our dynamic model, we decided to consider steady feet that are constantly in contact with the ground. This simplification gives us the possibility to remove 3 DoF but additional constraints to the optimization problem have to be introduced to guarantee that the effects of the feet-ground contact are correctly represented. Referring to Figure 3.6 three conditions have to be imposed:

- the unilaterality of the normal contact force

$$F_{n_{O_0}} > 0,$$

- the positioning of the contact force in the friction cone

$$\left| \frac{F_{t_{O_0}}}{F_{n_{O_0}}} \right| < \mu,$$

- the positioning of the contact actions center of pressure inside the contact area

$$-F_{n_{O_0}} a_{0,1} \leq u_1 \leq F_{n_{O_0}} a_{0,2}, \tag{3.2}$$

  where both $a_{0,1}$ ($= 0.200$ m) and $a_{0,2}$ ($= 0.058$ m) are greater than zero, whereas the lever of the tangential force is neglected because of the mono-dimensional model of the feet.

If these constraints hold, we have the guarantee that the removed feet are in a state of static equilibrium and the 5 DoF model correctly represents the whole body dynamics.

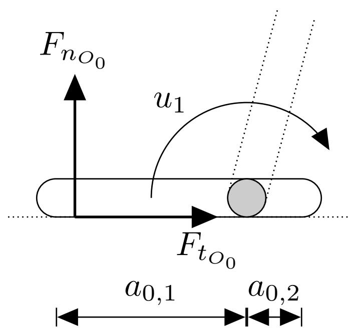


Figure 3.6: Feet free body diagram.

3.7 A Multibody Approach to the Definition of Dynamic Constraints

In this section we illustrate an idea that we tried to develop, but we put (temporarily) apart because of a lack of results. The idea consists in a different way to impose dynamic constraints (i.e. the ODE model) in Problem 2.2. Noticing the ease with which it's possible to introduce algebraic constraints and the extreme complexity of the equations of motion generated by such an articulated system, we thought that a "multibody approach" could make a difference in terms of performances.

Considering our dynamic system, what can be made is a decomposition of the chain in order to separate each one of the five links from the others. Defining the position and the orientation of each link with a redundant number of coordinates (x, y, θ) and introducing symbolic forces (as optimization variables) that represent the internal actions necessary to restore the congruence of the chain; we are able to analyze the dynamics of the whole system link by link. Looking at the equilibrium of the generic i^{th} body (see Figure 3.7), we can write a set of scalar equations

$$
\begin{aligned}
&m_i\ddot{x}_i = F_{i+1\to i,x} + F_{c_{O_i},x} + F_{i-1\to i,x},\\
&m_i\ddot{y}_i = m_i g + F_{i+1\to i,y} + F_{c_{O_i},y} + F_{i-1\to i,y},\\
&i_i\ddot{\theta}_i = u_i - u_{i+1} - (F_{i+1\to i,x} + F_{c_{O_i},x})C_iO_i\sin\theta_i + (F_{i+1\to i,y} + F_{c_{O_i},y})C_iO_i\cos\theta_i +\\
&\qquad + F_{i-1\to i,x}C_iO'_{i-1}\sin\theta_i - F_{i-1\to i,y}C_iO'_{i-1}\cos\theta_i,
\end{aligned}
$$

where O'_{i-1} is the point that belongs to the i^{th} link corresponding to O_{i-1} and (for example) $F_{i+1\to i,x}$ refers to the x component of the force that the $i+1^{th}$ link does on the i^{th} link.

Adding to these equations $2\cdot n$ constraints with the form

$$
\begin{aligned}
x_{i-1} + c_{i-1}\cos\theta_{i-1} &= xi - (a_i - c_i)\cos\theta_i,\\
y_{i-1} + c_{i-1}\sin\theta_{i-1} &= yi - (a_i - c_i)\sin\theta_i,
\end{aligned}
$$

we have a problem that is equivalent to the previous.

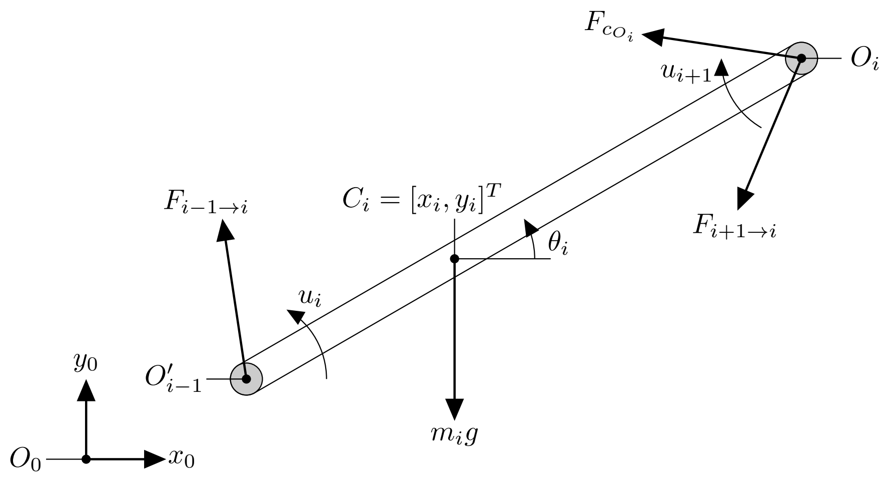


Figure 3.7: Single link free body diagram.

Approaching the optimization that way, we are defining a problem with $4 \cdot n \cdot N$ additional variables of optimization but whose dynamic equations are extremely simple (thanks to a constant diagonal inertia matrix and the absence of centrifugal and Coriolis terms) and whose Jacobian matrices, of the constraints that restore the congruence of the movements, are very sparse.

This fragmentation leads to a series of further simplification. Following the standard approach, the majority of the constraint we impose to the problem result in an high computational cost, taking as example the bounds imposed to the ankle hinge forces: using standard ODE models the computation of these values requires the calculation of terms such as the acceleration of the whole system center of mass (Equation 3.1) whereas, in this case, they simply are variables of optimization. On the other hand there are some drawbacks; for example the use of absolute angles makes the constraints that prevent self collisions more complicated to be expressed.

Speaking of optimization methods, we believe that this approach to introduce dynamic equations suits the direct collocation method very well. First of all the obtained problem is very sparse, fact that goes in the same direction of the collocation idea. Furthermore all the extra constraints we introduce have to be checked at every node (a division between the link, even for a short period of time, is not admissible) and thinking to the multiple shooting methods, this would require a constant dialogue with the integrator that at every integration step would have to check these equalities (whereas using a collocation method the grid on which the constraints are checked is the same used by the integrator).

As said this approach did not give us any results and we had not enough time to make a depth analysis of the problem; anyway in the light of the possible advantages that we have briefly highlighted here, the implementation of an efficient code using this technique is one of the next steps we want to take.

Chapter 4

Optimal Control Problem: Description and Results

In this chapter we present the results of several plannings. As said before the case study is a simplified humanoid and the optimization technique is the direct collocation method. In order to obtain different motion tasks, we combine two of three basic configurations (supine, standing, and prone) as initial and goal positions.

We start formalizing the optimal control problem: tasks, cost functions, constraints, and initial guesses are presented. In the second part some selected results are illustrated and commented.

All the problems we analyze are coded in Python language, using the interface to the open-source CasADi framework [16]. CasADi is a symbolic framework for algorithmic differentiation and numeric optimization, whose main purpose is to be a low-level tool for quick, yet highly efficient implementation of algorithms for nonlinear optimization. This great efficiency is achieved taking advantage of some state-of-art optimization routines (in our case the Ipopt [17] interior-point method is used).

4.1 Formalization of the Optimal Control Problem

In this section we start defining each element of the CT Optimal Control Problem 1.1 for the case study, after that the direct collocation method is used to derive a finite dimensional NLP.

4.1.1 Benchmark Tasks

At first, we sketched our problem considering a supine initial position and a standing goal posture, after we obtained first results we introduced an additional prone configuration. Starting from these three reference configurations (Figure 4.1), we defined the following planning problems: prone to standing, standing to prone, supine to standing, and standing to supine. These tasks are thought to highlight the principal difficulties that the solver might have to face. Analyzing the first case, the linear interpolation we give as initial guess (Section 4.1.4) is not a feasible movement because of the ankle torque constraint (Equation 3.2) anyway, since the feet frontal length $a_{0,1}$ is sufficiently high, this lower bound is not too strict. This gives the possibility to find a feasible trajectory (in a very few number of iterations) that is extremely far away from the optimum described by the cost function, in this sense the objective function will have to guide the optimization to a vast exploration of $\mathcal{C}$. In case of a supine initial position the interpolation represent an highly unfeasible trajectory (note that $a_{0,2} \ll a_{0,1}$) and this time the algorithm will have to deal with a "one-way-out" search in $\mathcal{C}$. Generally we expect that the solver will ride out of these difficulties by and extended hands-ground interaction, for this reason we propose the two remaining tasks: in addition to the previous complications, these times the humanoid starts its motion with the hands separated from the ground and it will have to understand itself the usefulness of possible interactions with the environment.

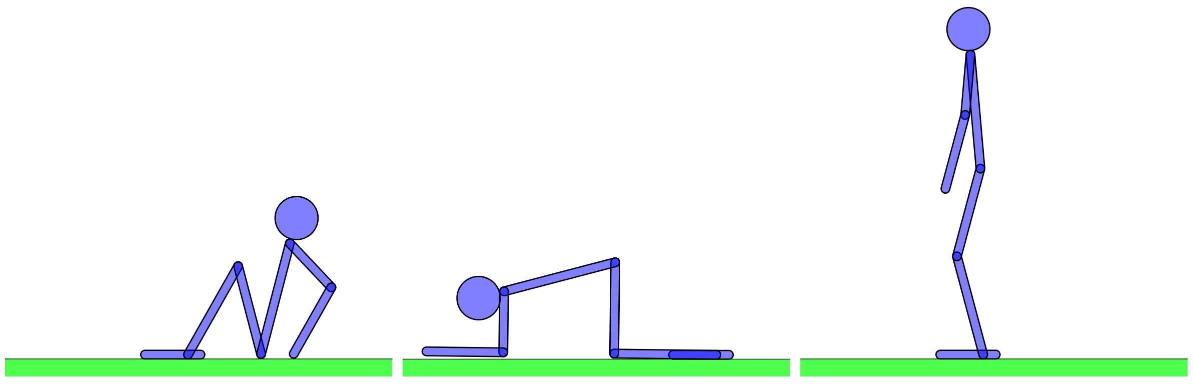

(a) Supine. (b) Prone. (c) Standing.

Figure 4.1: Different initial and goal configurations.

It is important to relate an issue that comes out imposing initial and goal conditions to this kind of problems. Since contact force models are smooth, if particular bounds are imposed to the input torques (such as the ones described in Section 3.6), it is practically impossible that the configurations we define are equilibrated. For example if we want to plan a motion that starts from the supine posture (Figure 4.1a) it is logical to compute configuration angles such that the hips and the hands of the humanoid are exactly placed on the floor. If for simplicity we think to do not have the first joint actuated, looking at the entire system, it is evident that equilibrium is achieved only with hips and hands penetration/separation in/from the ground. Furthermore things can become even more complicated if some other bounds have to hold (for example the non-negativity of the force applied by the ground on the feet). To deal with this kinds of problems we implemented a loop that, acting on the heel angle q_1, numerically search the nearest feasible configuration to the one introduced. In light of this, real initial and goal conditions can be imperceptibly different ($\approx 10^{-2}$ rad thanks to the high values of stiffness) from their nominal values.

4.1.2 Cost Function

In practical optimal control problems multiple objective functions are almost inevitable to introduce: most of the time a simple cost function where, for example, the norm of the inputs is minimized is not sufficient to achieve complex motion behaviours. On the other hand it's important to keep to the minimum the number of cost terms, in fact the introduction of too many weights can make the relation between cost functions and results very difficult to interpret.

Multi-Objective Optimization (MOO) is a wide field and analyzing these aspects is beyond the goals of this study, anyway it is important be aware of the main difficulties that can arise using multiple cost functions. For this reason in the following lines we introduce, in very general terms, the fundamentals features of this area.

Denoting the objective function as $\phi(v) = \phi_1(v) + \phi_2(v) + ... + \phi_{n_\phi}(v)$, it is possible to encircle the subspace of $\mathbb{R}^{n_v}$ where all the values of v that respect the constraints live (*feasible region*); mapping this subspace through $\phi(v)$ to $\mathbb{R}^{n_\phi}$ we can delimit the *feasible objective region* (note that a weighted sum of the cost terms is represented in this space as a set of parallel hyperplane that intersect each axis in a non-negative point). A solution of the MOO problem has to lie on boundary of the feasible objective region, in particular we define the *Pareto front* the space $\in \mathbb{R}^{n_\phi - 1}$ that contains all of the trade-off solutions (for which it is impossible to reduce one objective further without necessarily increasing one of the others). In case of a weighted sum of cost terms, this solution will be the point where the nearest to the origin hyperplane (among those which cross the feasible region) intersect the Pareto front. Pareto fronts can be convex, non-convex or even disconnected (with increasing difficulty in solving the associated MOO problem).

In general two aspects are helpful approaching these kind of problems:

- an homogeneous scaling of each cost term (for example, for the i^{th} term $\phi_i^{sc}(v) = \frac{\phi_i(v) - \phi_i^{min}}{\phi_i^{max} - \phi_i^{min}}$) avoids too thin feasible regions (that can make the relation between the cost function weights and the respective non-dominate solution ill-conditioned);

- the use of different (from a weighted sum) functions to aggregate cost terms leads to more complex hypersurfaces (than the hyperplanes) that can be able to find solutions of the MOO even inside the possible non-convex regions of the Pareto front.

Coming back to our problem, we decided to use an objective function composed by four Lagrangian terms, in particular we decided to penalize inputs torques, angular velocity, angular accelerations, and the distance of the hands from the ground. The weights adopted to penalize each term are indicated by the letter W, in the specific case of the inputs we decided to introduce a diagonal matrix of weights (as opposed to a scalar) since it gave us the possibility to finely tune the humanoid behaviours. The cost function terms are summarized in the following list:

- *inputs penalization*

$$L_u = \int_0^T u(t)^T W_u u(t) dt \approx \sum_{i=0}^{N-1} u_i^T W_u u_i \Delta t;$$

- *velocities penalization*

$$L_{\dot{q}} = W_{\dot{q}} \int_0^T \dot{q}(t)^T \dot{q}(t) dt \approx W_{\dot{q}} \sum_{i=0}^{N} \dot{q}_i^T \dot{q}_i \Delta t;$$

- *accelerations penalization*

$$L_{\ddot{q}} = W_{\ddot{q}} \int_0^T \ddot{q}(t)^T \ddot{q}(t) dt \approx W_{\dot{q}} \sum_{i=0}^{N-1} (\dot{q}_{i+1} - \dot{q}_i)^T (\dot{q}_{i+1} - \dot{q}_i) \frac{1}{\Delta t};$$

- *hands distance from the ground penalization*

$$L_{y_H} = W_H \int_0^T y_H(t)^2 \left(1 - \frac{t}{T}\right) dt \approx W_H \sum_{i=0}^{N} y_{H_i}^2 \left(1 - \frac{i}{N}\right) \Delta t.$$

Unlike the first three cost terms, the penalization of the distance of the hands from the ground is quite unusual. Its presence is due to the fact that in almost every stand-up or sit-down movement it is convenient to quickly approach hands to the ground (in order to reach a more stable position and share forces between limbs more efficiently). The time-dependent coefficient is introduced to relax this “soft constraint” as time goes on.

Since the terms of the weight matrices change from a planning to the other, they will be specified case by case.

4.1.3 Constraints and Variables Bounds

Using using Ipopt, constraints have to be expressed as inequalities, this can be easily done by setting the lower and the upper bounds of the residuals (related to the expression of the constraint we want to introduce) coincident to zero. Implying this, in the following lines we will define all the constraints we impose to our motion planning problem as equalities.

The first condition to impose is the validity of the ODE model at each collocation node, the discretization scheme we exploit is the *Mid-Point Rule* (MPR). Considering the generic i^{th} node for $i \in \{0, 1, ..., N-1\}$, we have

$$\begin{aligned}
&B\left(q_{i+\frac{1}{2}}\right)\frac{\dot{q}_{i+1}-\dot{q}_i}{\Delta t} + C\left(q_{i+\frac{1}{2}}, \dot{q}_{i+\frac{1}{2}}\right)\dot{q}_{i+\frac{1}{2}} + G\left(q_{i+\frac{1}{2}}\right) - Q\left(q_{i+\frac{1}{2}}, \dot{q}_{i+\frac{1}{2}}, u_i\right) = 0;\\
&q_{i+1} - q_i - \dot{q}_{i+\frac{1}{2}}\Delta t = 0;
\end{aligned}$$

where $q_{i+\frac{1}{2}} = \frac{q_{i+1}+q_i}{2}$ and $\dot{q}_{i+\frac{1}{2}} = \frac{\dot{q}_{i+1}+\dot{q}_i}{2}$. As we will see in Section 5.2.1, sometimes we will eliminate less significant terms from the dynamic equations in order to obtain a fast approximated solution to the problem; in these cases the procedure illustrated above is exactly the same (except for the ignored terms).

Another fundamental set of constraints we impose is the one necessary to restore the effects of the removed feet, in this case the inequalities we impose are the ones presented in Section 3.6. From a practical point of view, the only difficulty that comes out is the computation of the constraint force made by the fixed hinge. Since we derived equations of motion using an energy method, this value is not available and its expression has to be calculated considering the equilibrium of the entire system

$$F_{c_{O_0}} = m(a_C - g) - \sum_{i=1}^{n} F_{c_{O_i}},$$

note that in order to be coherent with the MPR, this equation has to be discretized as

$$F_{c_{O_0}}(q, \dot{q}, \ddot{q}) \approx F_{c_{O_0}}\left(q_{i+\frac{1}{2}}, \dot{q}_{i+\frac{1}{2}}, \frac{\dot{q}_{i+1}-\dot{q}_i}{\Delta t}\right).$$

In Section 5.2.2 we will analyze the computational differences between an explicit introduction of the contact forces model in the equations of motion and the definition of these forces as optimization variables. In case the second alternative is preferred, a set of equalities (that force these addition variables to be equal to their expressions as functions of positions and velocities) has to be introduced. This is done simply employing relations presented in Section 3.4. Once again, in order to be coherent with the MPR we have to perform the discretization

$$F_{c_{O_i}}(q, \dot{q}) \approx F_{c_{O_i}}(q_{i+\frac{1}{2}}, \dot{q}_{i+\frac{1}{2}}).$$

Thanks to the two-dimensional structure of our example, we can guarantee the absence of self collisions simply by imposing bounds to the configuration. In Table 4.1 we list the values of these bounds.

Angle number	**Lower bound**	**Upper bound**
1 (ankles)	-90°	270°
2 (knees)	190°	360°
3 (hips)	0°	170°
4 (shoulders)	0°	270°
5 (elbows)	190°	360°

Table 4.1: Bounds for the configuration vector.

4.1.4 Initial Guess

The last topic to discuss, in order to complete the overview of the in case optimal control problem, is the initial guess. Considering the complexity of the movements we plan, find a simple and generally valid initial guess is not an easy task. We tried different alternatives and, as regard configuration and velocities, our decision ended to be a linear interpolation (of the configuration relative angles) between the initial and the goal conditions. In order to have an idea of what kind of movement this is, we illustrate an example in Figure 4.2, as it can be seen this is quite different from an human-like motion but it demonstrated to work effectively.

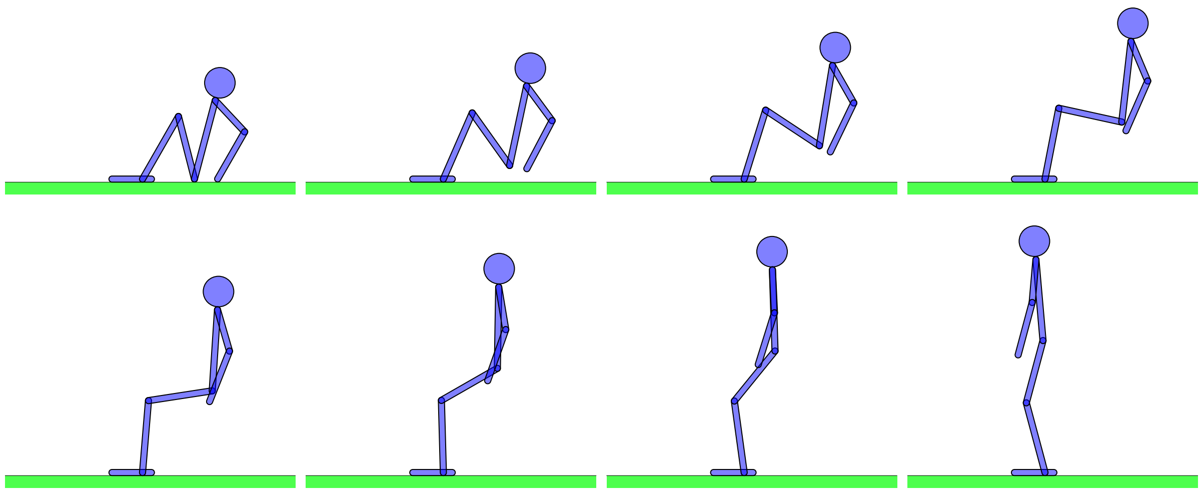


Figure 4.2: Linear interpolation between supine and standing postures.

For what concerns the inputs initial guess, our first try was a computation of the torques necessary to perform the interpolation movement but this approach didn't gave good results. In fact, looking at Figure 4.2, it's clear how the inputs needed by such a trajectory are extremely high while the results we expect to obtain go in the opposite direction. Furthermore, in this case the heels actuator would require a great effort ($u_1 \gg 0$) whereas, as exposed in Section 3.6, we constrained this value to be almost zero. In light of these observations, we decided to initialize the input torques with a constant value equal to 0.

In Section 5.2.2 we will discuss the possibility to define contact forces as optimization variables, in this cases we will have the need to also define an initialization for these additional variables. Unlike the previous case, this time the computation of contact forces consequent with the interpolation movement worked well.

4.2 Reference Results

In this section we present the results of the motion planning problems we described above. Referring to the techniques we will present in Chapter 5, all the results we propose are obtained using the pseudo-static intermediate solution method (presented in Section 5.2.1) and by considering the contact forces as variables of optimization (Section 5.2.2). Each one of the following results is obtained considering a planning time T equal to 5 s and 100 collocation nodes N. Times and numbers of iterations required to compute these trajectories will be discussed in Section 5.2.3.

4.2.1 Prone to Standing

The first rising motion we present starts from the prone configuration (Figure 4.1b) and finishes when the standing position (Figure 4.1c) is reached. This is probably the simplest movement we can think to: the hands are in contact with the ground from the very first moment and the ankle torque bound doesn't represent a particular difficulty ($a_{0,1} \gg a_{0,2}$).

Considering the nature of this movement, we decided to introduce in the dynamic equations only the contact forces that are applied on: the hands, the elbows, and the knees. Because of that the total number of optimization variables is equal to

$$\underbrace{2 \cdot n \cdot (N+1)}_{\text{states}} + \underbrace{n \cdot N}_{\text{inputs}} + \underbrace{2 \cdot 3 \cdot N}_{\text{contact forces}} = 2110.$$

In this case we considered the following values for the weights of the cost function:

$$\begin{aligned} W_u &= \operatorname{diag}(0.100, 0.010, 0.018, 0.002, 0.002), \\ W_{\dot{q}} &= 100, \\ W_{\ddot{q}} &= 100, \\ W_H &= 1000. \end{aligned}$$

In Figure 4.3 we illustrate the obtained dynamic motion.

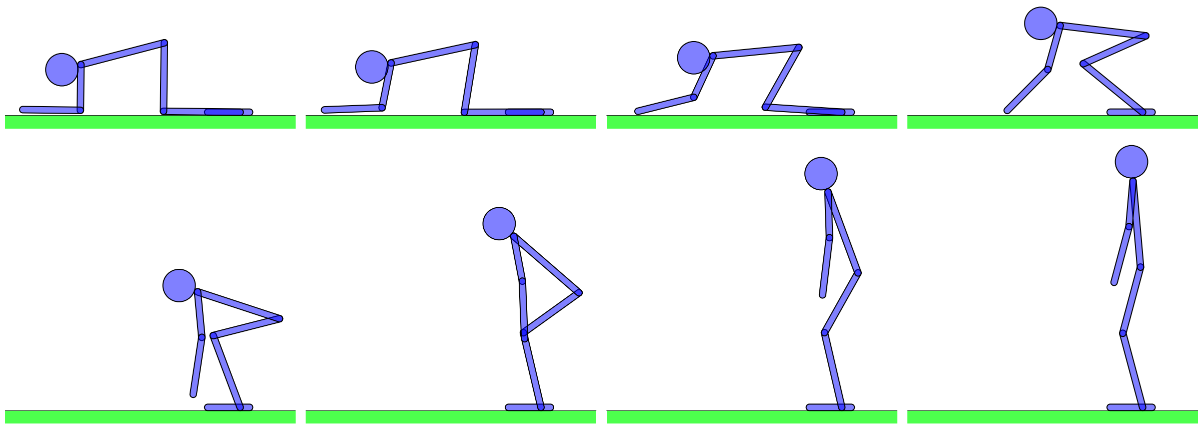


Figure 4.3: Optimal trajectory for the “prone to standing” problem; 0.71 s per frame.

In this case the initial position required some adjustments to respect each constraint. In particular, the rotational equilibrium about the ankle axis of the contact and weight forces, leads to a rotation about the same axis of -0.012 rad ($= -0.69°$).

In Figure 4.4 the evolution of the states and the inputs can be observed. As can be seen the obtained movements are very smooth thanks to the strong penalization imposed to the accelerations. Even if there aren’t bounds to the inputs, we believe that the computed values of torques are very likely: in particular the joints that require the highest values of torque are the knee and the shoulder (in both cases an maximum input almost equal to 200 Nm is needed).

As it can be seen in Figure 4.5, the constraints on the heel force are verified: the normal contact force never holds the humanoid to the ground, the absolute value of the ratio between tangential and normal force is always less than the friction coefficient $\mu = 1$, and the center of pressure of the action applied to the feet lies in the contact area $[-a_{0,1}, a_{0,2}] = [-0.200, 0.058]$ m.

A very significant result to analyze is the value of the contact forces applied on the hands by the ground (Figure 4.6): during the central phase of the movement, the the load applied to the arms is almost a third of the total wight of the humanoid (696 N). Note that, since the friction coefficient is equal to 1, during a sliding phases the normal and the tangential forces have the same absolute values and the related graph is symmetrical (or overlapped).

At last we present the percentages of the total cost to ascribe to each term of the objective function (Figure 4.7). As it can be seen inputs torques are penalized a little more than the other terms and, in spite of the shoulder input reaches very high values,

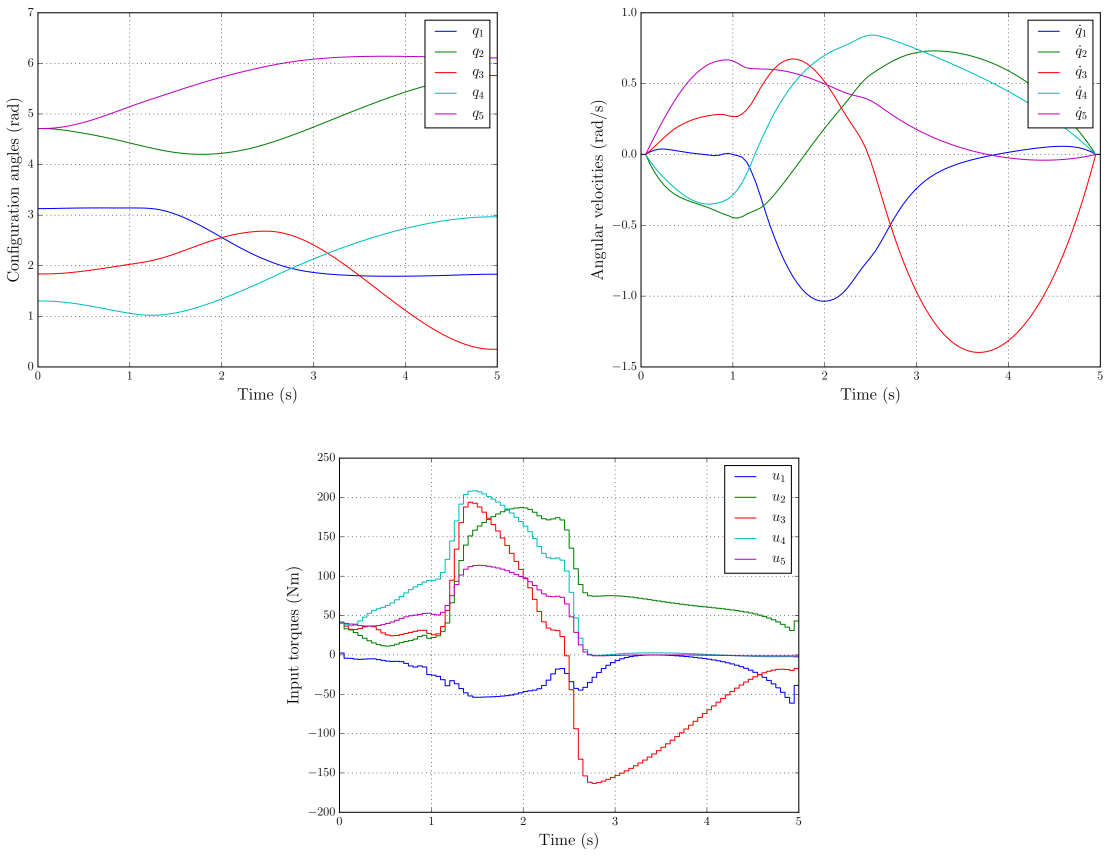


Figure 4.4: States and inputs as a function of time; "prone to standing" case.

the related percentage of the total cost is relatively small because of its low weight in the objective function.

4.2.2 Standing to Prone

The second result we present is the reverse of the first case: starting from a standing position (Figure 4.1c), the humanoid has to plan a movement to reach a prone configuration (Figure 4.1b). The main difficulty is represented by the extreme stiffness of the contacts; in fact, in order to reduce the inputs, a precise use of the hands is necessary but the gradient of the normal force, computed at the initial position of the hands (0.77 m), is practically equal to zero. Comparing this case with the previous one, we expect to obtain a greater influence on the final result of the cost term related to the distance of the hands from the ground (since its integrand function is inversely proportional to time).

Even this time we only considered the contact forces that are applied on: hands, elbows, and knees. In light of this the total number of optimization variables is equal to

$$\underbrace{2 \cdot n \cdot (N+1)}_{\text{states}} + \underbrace{n \cdot N}_{\text{inputs}} + \underbrace{2 \cdot 3 \cdot N}_{\text{contact forces}} = 2110.$$

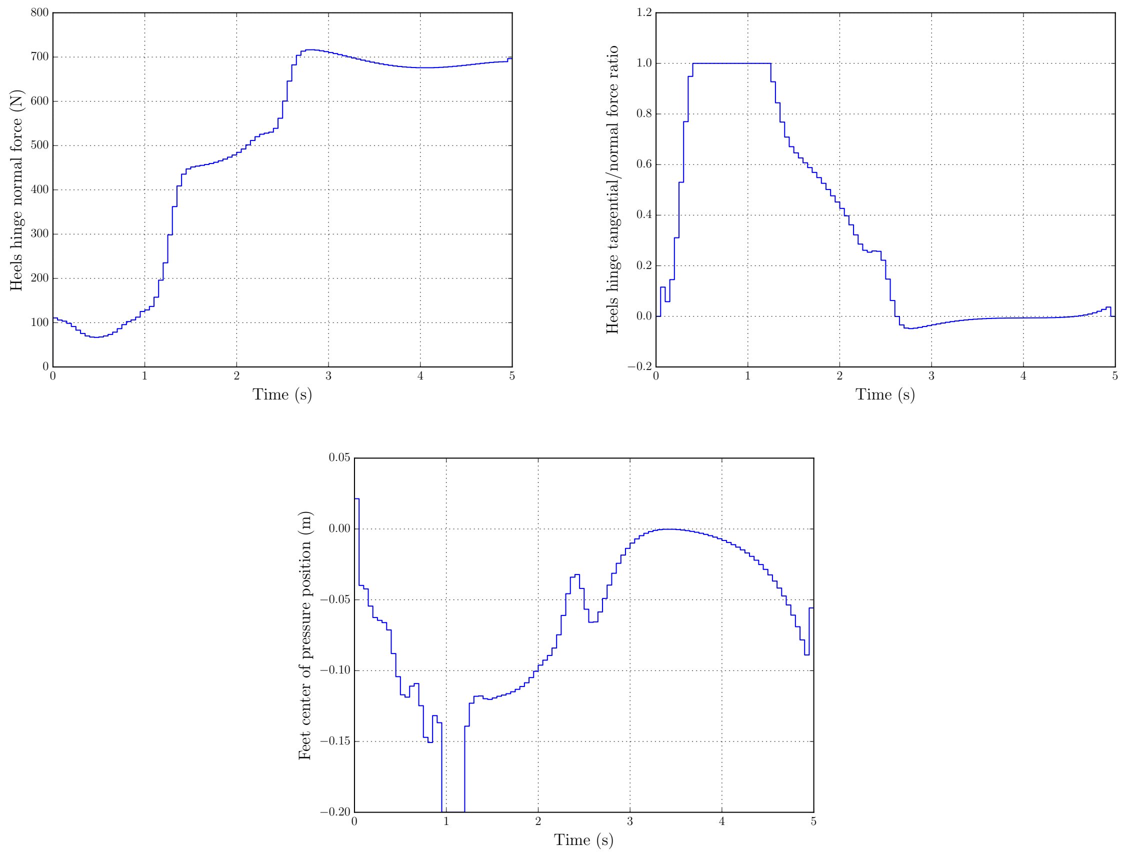


Figure 4.5: Heels constraint parameters as a function of time; “prone to standing” case.

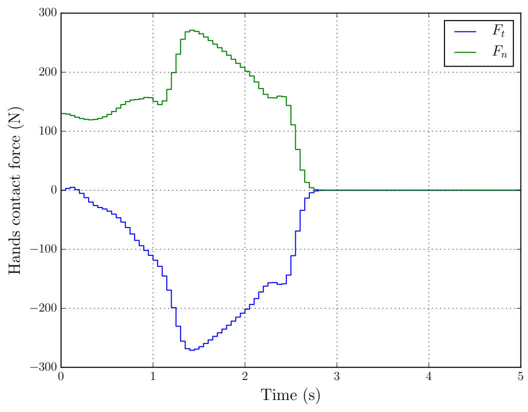


Figure 4.6: Normal and tangential contact forces applied on the hands, originated from the interaction with the ground; “prone to standing” case.

We considered the following values for the weights of the objective function:

$$\begin{aligned} W_u &= \text{diag}(0.030, 0.015, 0.060, 0.003, 0.003), \\ W_{\dot{q}} &= 50, \\ W_{\ddot{q}} &= 50, \\ W_H &= 5000. \end{aligned}$$

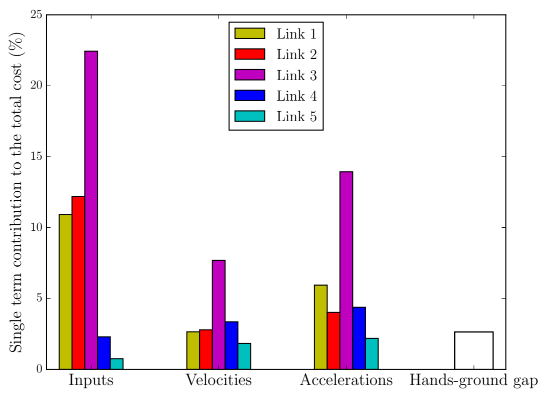


Figure 4.7: Percentages of the total cost related to each term of the objective function; “prone to standing” case.

In Figure 4.8 the result of the planning is illustrated.

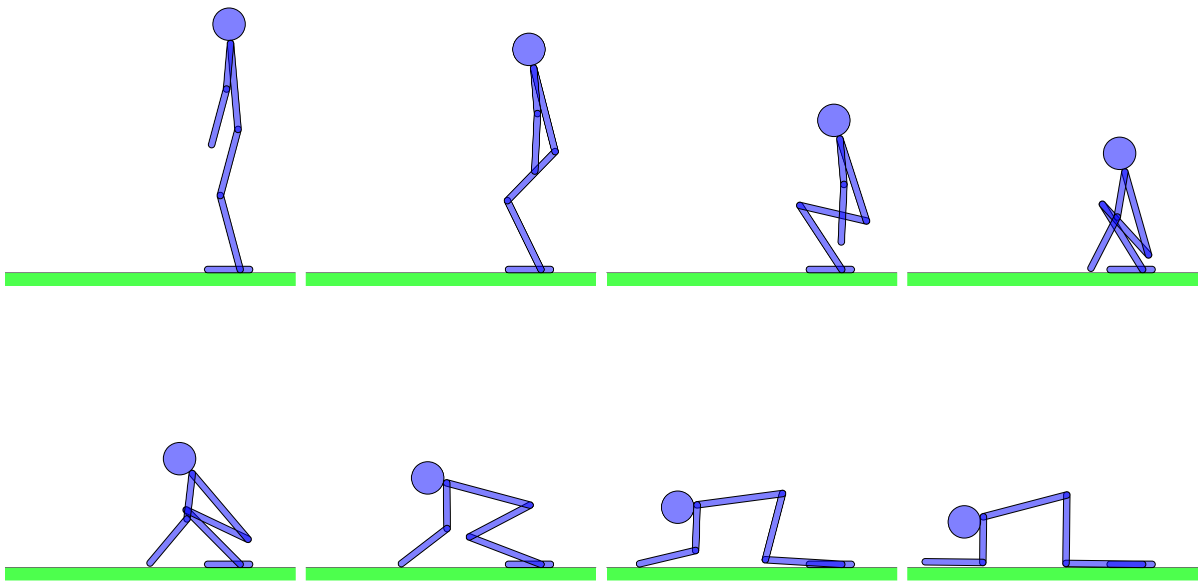


Figure 4.8: Optimal trajectory for the “standing to prone” problem; 0.71 s per frame.

Since this motion is opposite to the previous one, this time the adjustment (of $-0.69°$) is required by the goal position, while the initial configuration correspond to the nominal one.

In Figure 4.9 the evolution of states and inputs is presented. This time torques are even smaller (since the inertial forces are in the opposite direction of the gravity acceleration). The greatest input observed is applied by the knee actuator during the descent phase (as it might have been expected looking at Figure 4.8).

In Figure 4.10 we illustrate the parameters associated to the constraint force applied on the heels of the humanoid, even this time the bounds on these values are satisfied.

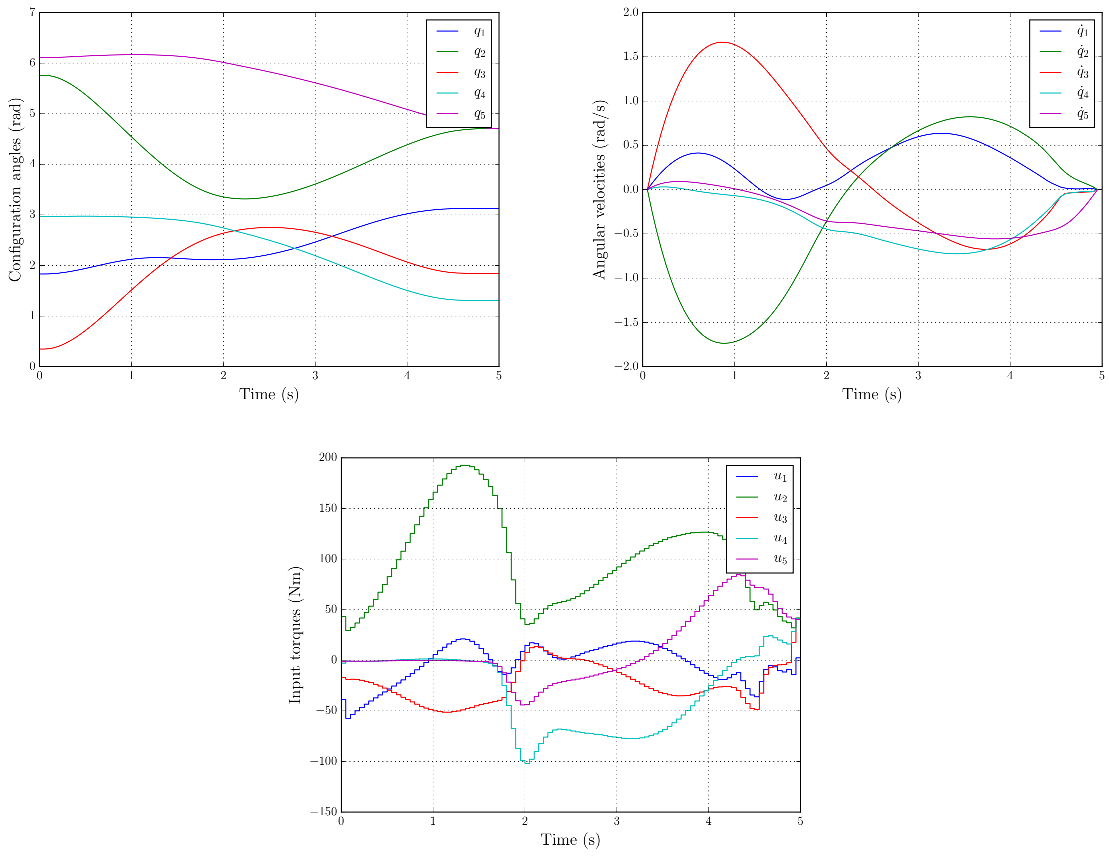


Figure 4.9: States and inputs as a function of time; "standing to prone" case.

The contact of the hands with the ground consists almost uniquely in a sliding phase, as it can be seen in Figure 4.11. As soon as the interaction starts, the humanoid begins its movement in the direction of the goal position.

To conclude we present the percentages of the total cost to ascribe to each term of the objective function (Figure 4.12). As pointed out, in this case the main cost is attributable to the distance of the hands from the ground.

4.2.3 Supine to Standing

In this section we present the first motion planning problem we approached: from supine (Figure 4.1a) to standing configuration (Figure 4.1c). Unlike previous tasks, in this case the ankle torque bound represent a great obstacle and complex hands-ground interactions are required to reach the goal posture.

This time we only introduced the contact forces that are applied on the hands and the hips, because of that the total number of optimization variables is equal to

$$\underbrace{2 \cdot n \cdot (N+1)}_{\text{states}} + \underbrace{n \cdot N}_{\text{inputs}} + \underbrace{2 \cdot 2 \cdot N}_{\text{contact forces}} = 1910.$$

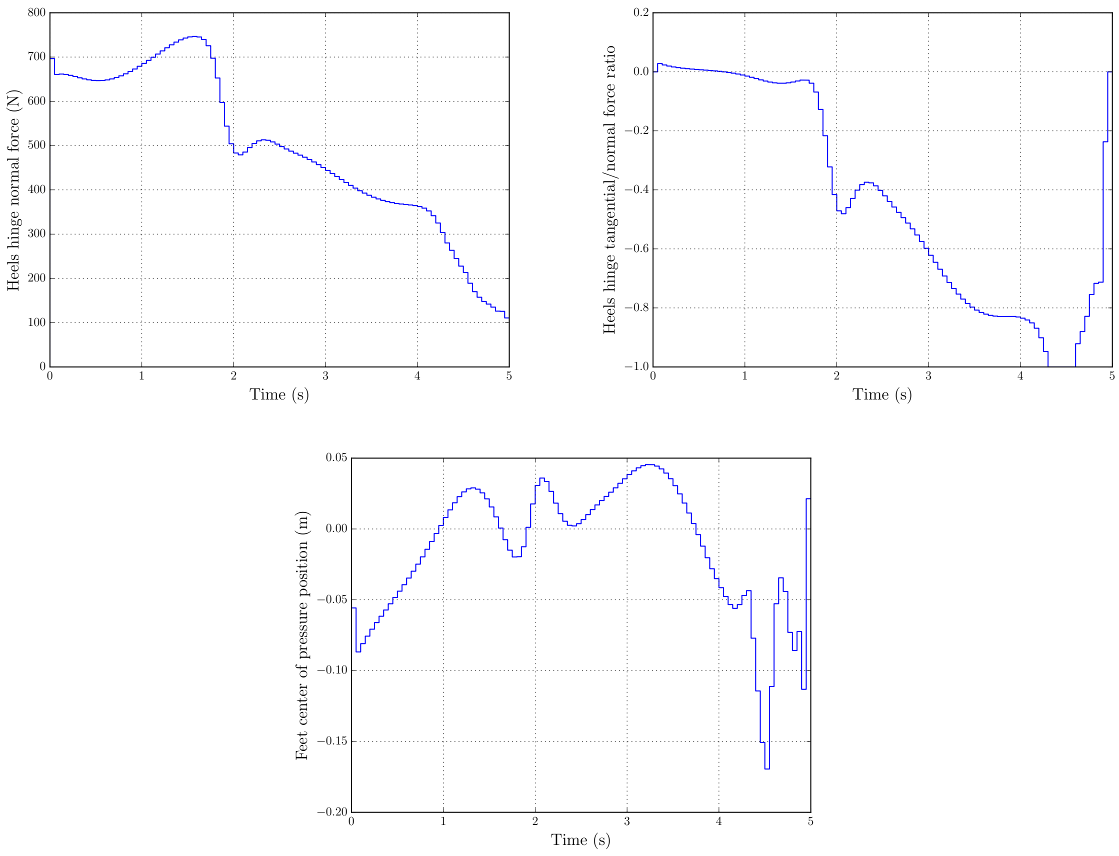


Figure 4.10: Heels constraint parameters as a function of time; “standing to prone” case.

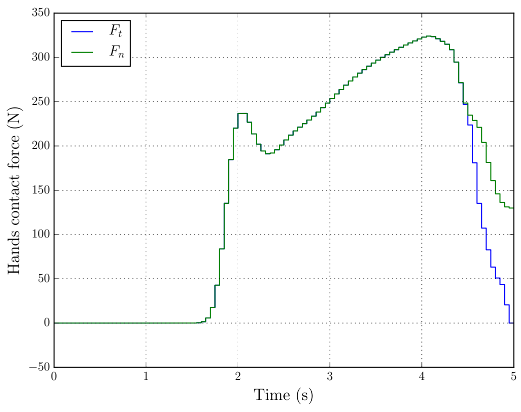


Figure 4.11: Normal and tangential contact forces applied on the hands, originated from the interaction with the ground; “standing to prone” case.

In this case the objective function was tuned as

$$
\begin{aligned}
W_u &= \text{diag}(0.10, 0.05, 0.05, 0.01, 0.01),\\
W_{\dot{q}} &= 50,\\
W_{\ddot{q}} &= 10,\\
W_H &= 1000,
\end{aligned}
$$

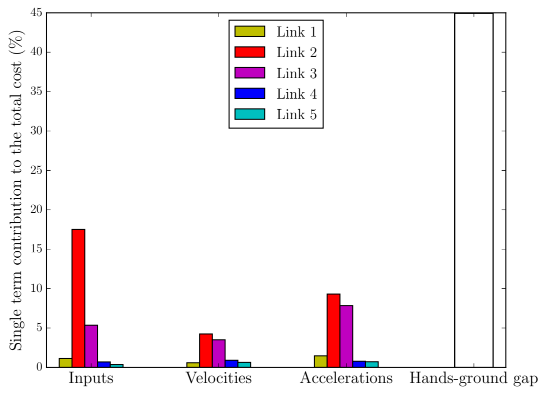


Figure 4.12: Percentages of the total cost related to each term of the objective function; “standing to prone” case.

note that the weight associated to accelerations is reduced from the previous cases, this in order to limit the smoothness of the movement and let the humanoid free to quickly use its hands.

Figure 4.13 represents the evolution of the humanoid configuration. To reach feasibility, a rotation of the initial position of 0.0074 rad ($= 0.42°$) about the heel axis is needed, while the nominal goal position is already feasible.

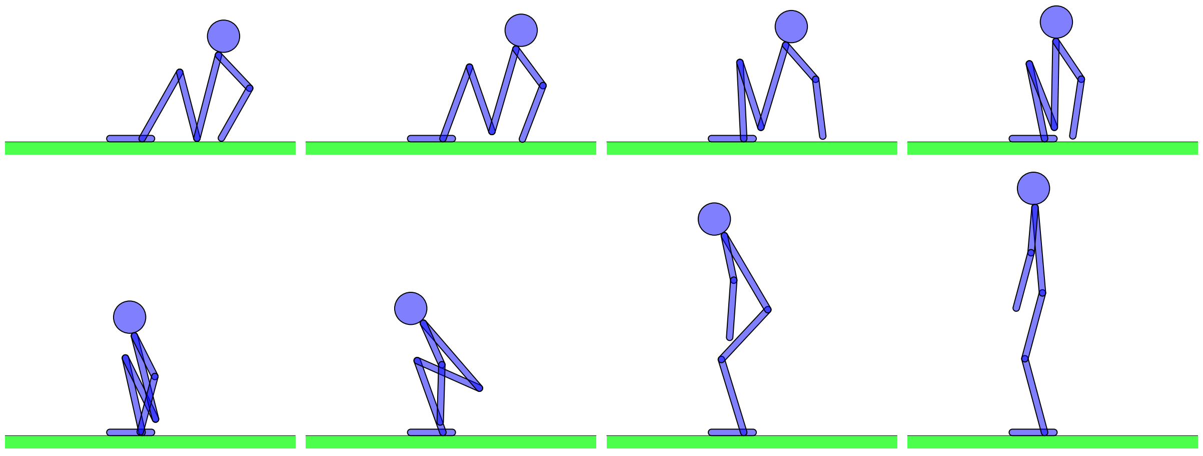


Figure 4.13: Optimal trajectory for the “supine to standing” problem; 0.71 s per frame.

In Figure 4.14 the evolution of the states and the inputs is presented. As it can be seen, an high value of torque has to be applied by the shoulder and the elbow actuators in order to contrast the body weight during the sliding phase, whereas in the second phase of the movement the prevailing inputs are the ones acted by the knee and the hip joints.

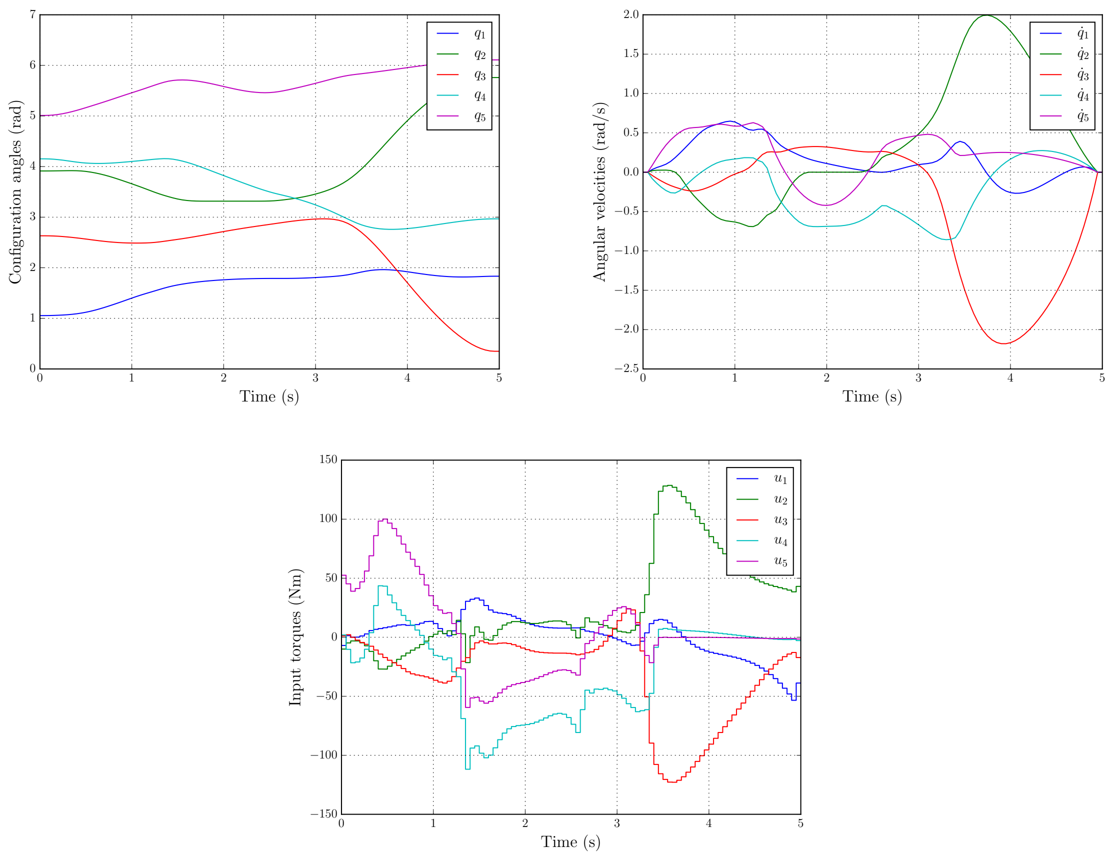


Figure 4.14: States and inputs as a function of time; “supine to standing” case.

In Figure 4.15 we present the parameters associated to the constraint force applied on the heels of the humanoid.

Figure 4.16 represents the hands-ground interaction. In this case three phases of the contact are noticeable: the first second and a half is needed to approach the hips to the feet and the hands don’t slip, in the central phase of the movement the hands slide on the ground, at the end of the contact an impulsive force in applied in order to ease the rising movement. This particular behaviour is obtained thanks to the reduction of the accelerations weight.

To conclude we present the percentages of the total cost related to each term of the objective function (Figure 4.17). This time the input costs are predominant.

4.2.4 Standing to Supine

The last motion we analyzed it the standing (Figure 4.1a) to supine (Figure 4.1c) movement. In this case the ankles torque bound is extremely limiting and the hands starts in a position very distant from the ground. Despite of these complications, even this time the solution obtained is very likely.

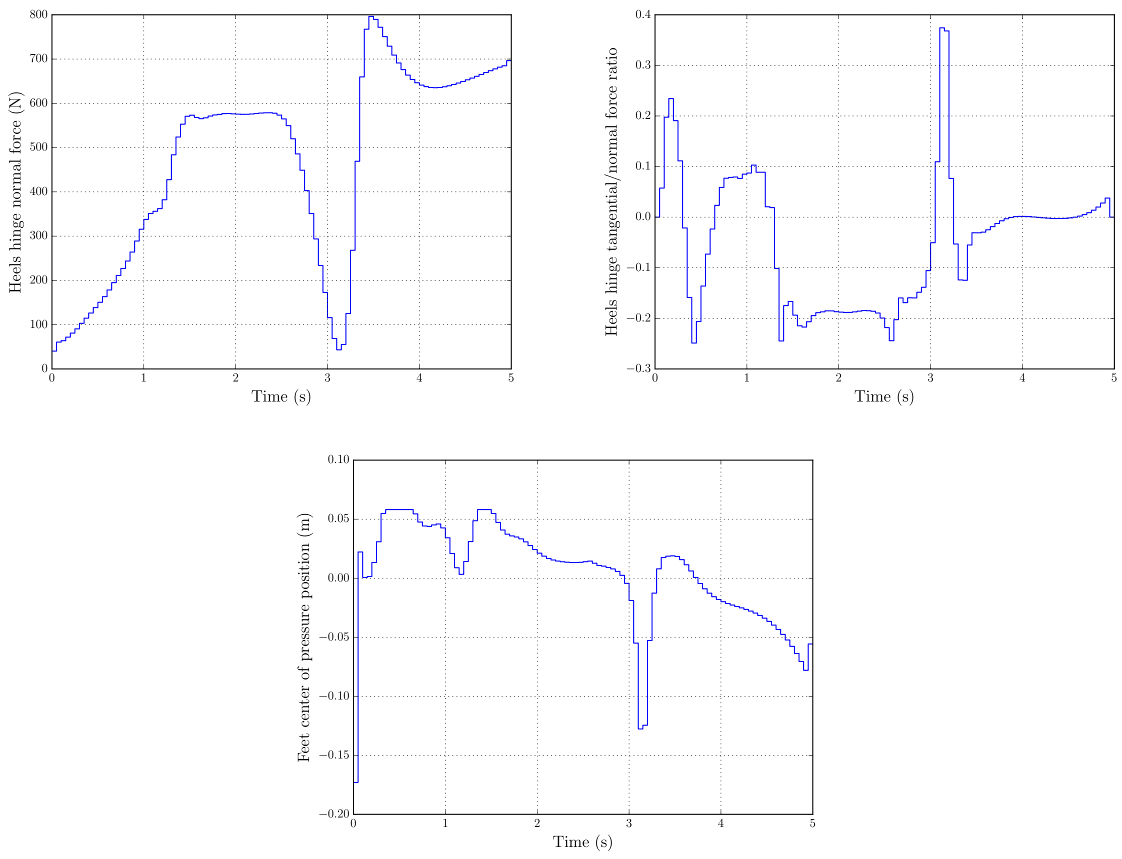


Figure 4.15: Heels constraint parameters as a function of time; “supine to standing” case.

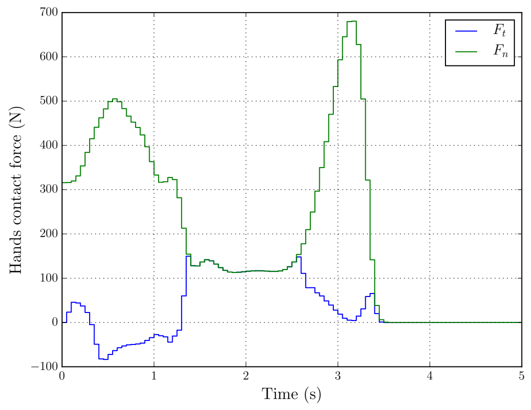


Figure 4.16: Normal and tangential contact forces applied on the hands, originated from the interaction with the ground; “supine to standing” case.

As in the previous case, we introduced only the contact forces that are applied on the hands and the hips and the total number of optimization variables is still equal to

$$\underbrace{2 \cdot n \cdot (N+1)}_{\text{states}} + \underbrace{n \cdot N}_{\text{inputs}} + \underbrace{2 \cdot 2 \cdot N}_{\text{contact forces}} = 1910.$$

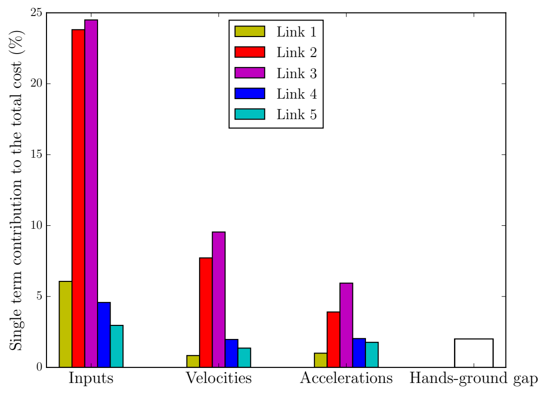


Figure 4.17: Percentages of the total cost related to each term of the objective function; "supine to standing" case.

The weights of the objective function used this time are

$$
\begin{aligned}
W_u &= \operatorname{diag}(0.10, 0.05, 0.05, 0.01, 0.01),\\
W_{\dot{q}} &= 50,\\
W_{\ddot{q}} &= 100,\\
W_H &= 1000.
\end{aligned}
$$

Figure 4.18 represents the evolution of the configuration of the humanoid. To reach feasibility a rotation of the goal position of 0.42° about the heel axis is needed.

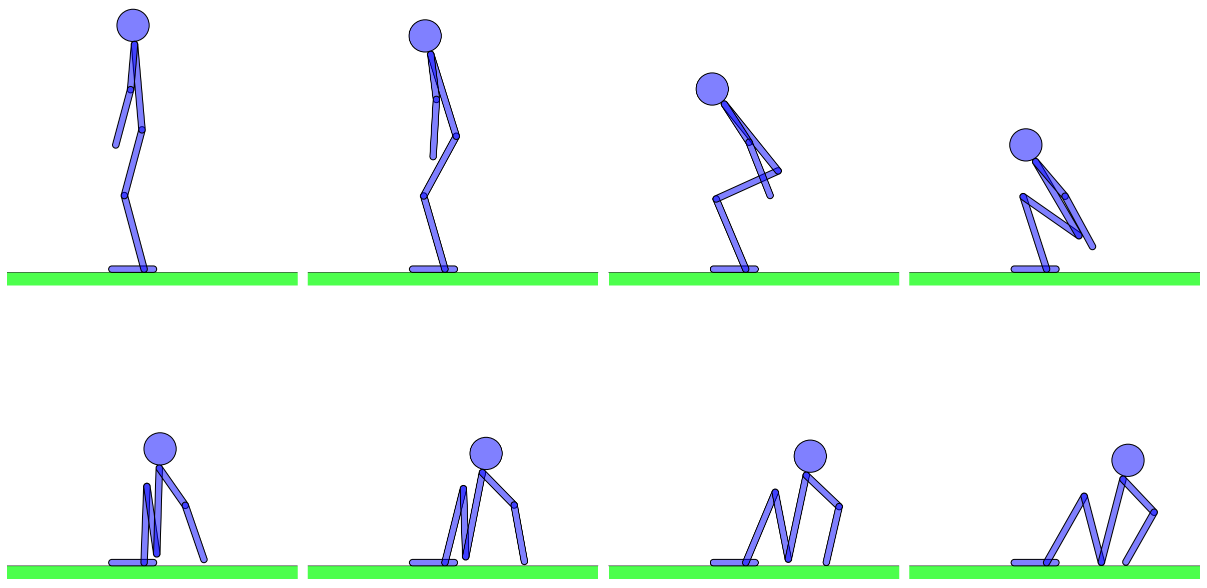


Figure 4.18: Optimal trajectory for the "standing to supine" problem; 0.71 s per frame.

In Figure 4.19 the evolution of the states and the inputs is presented. In this case, since the humanoid has to sit down, the input torques are generally lower than the case before.

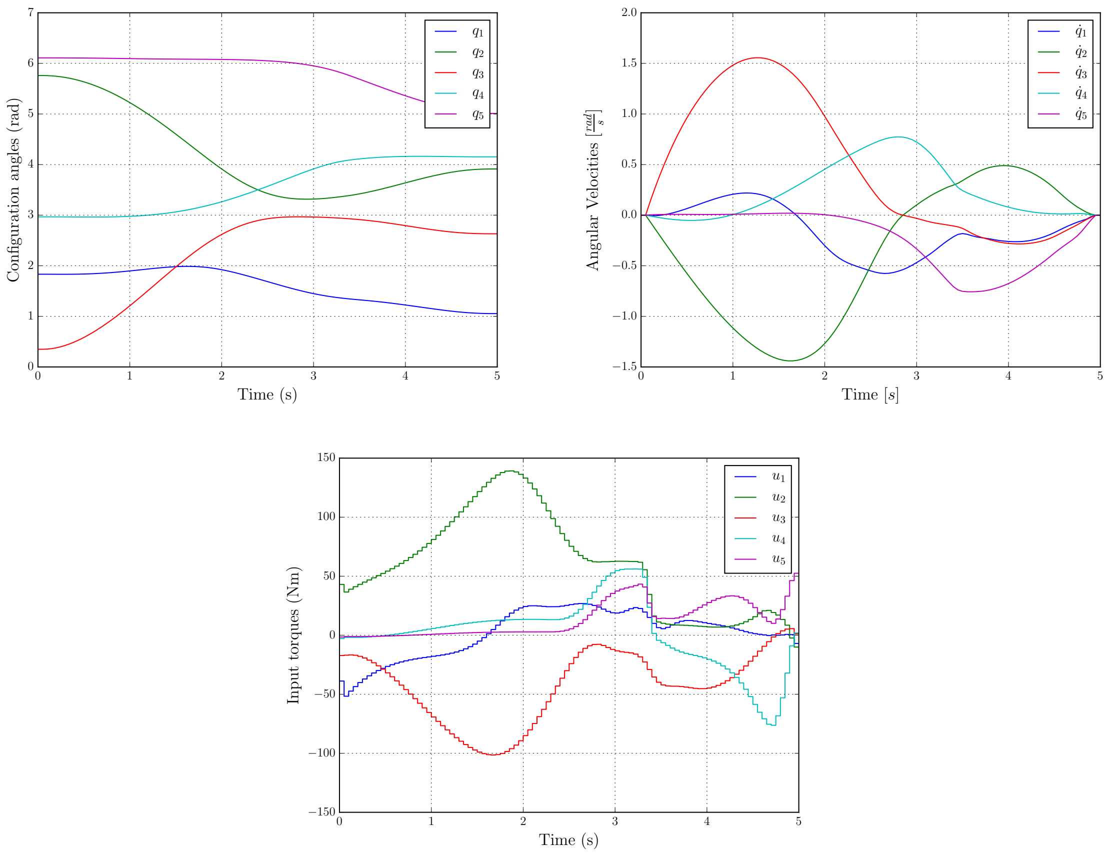


Figure 4.19: States and inputs as a function of time; "standing to supine" case.

In Figure 4.20 we present the parameters associated to the constraint force applied on the heels of the humanoid.

Figure 4.21 represents the hands-ground contact forces. In this case we can note that the sliding phase has a reduced duration if compared to the previous cases.

Eventually we present the percentages of the total cost related to each term of the cost function (Figure 4.22). This time the knee torque is the main cost of the movement, as it is logical to be looking at Figure 4.18.

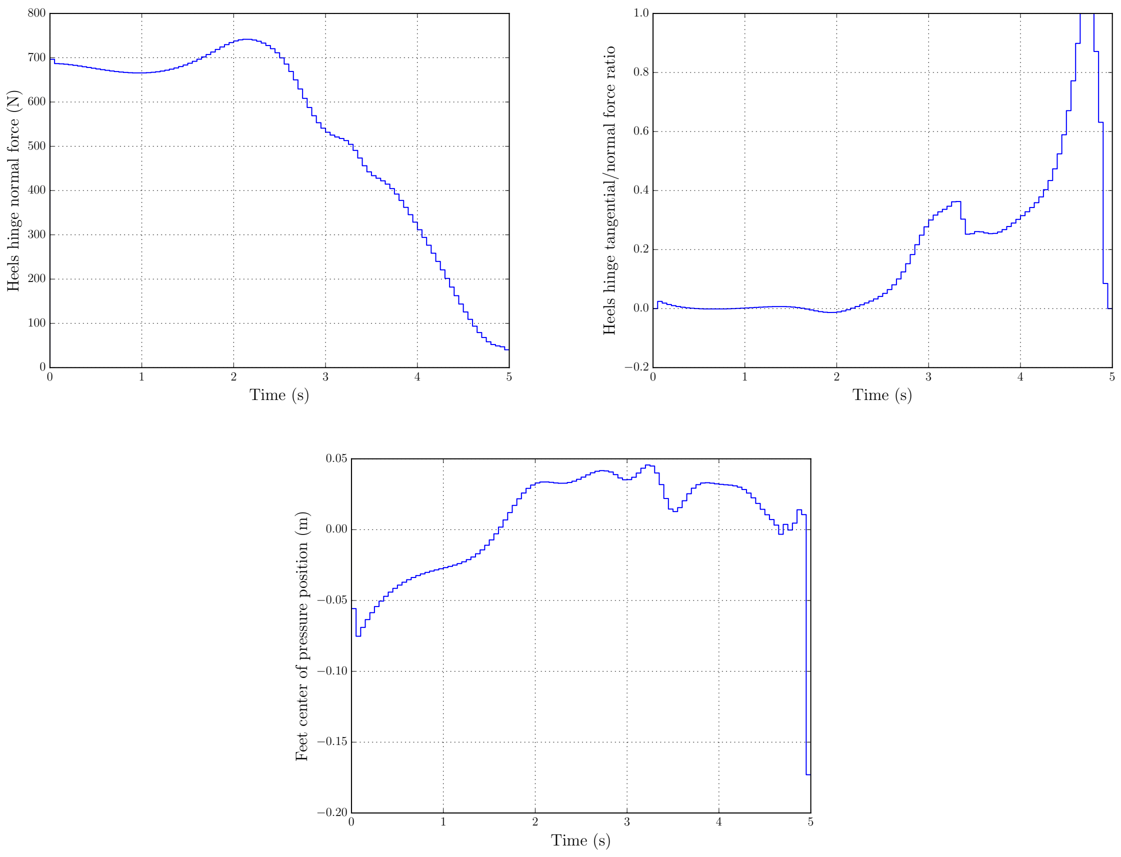


Figure 4.20: Heels constraint parameters as a function of time; “standing to supine” case.

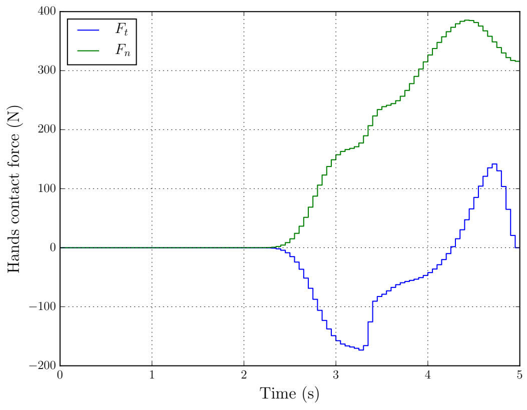


Figure 4.21: Normal and tangential contact forces applied on the hands, originated from the interaction with the ground; “standing to supine” case.

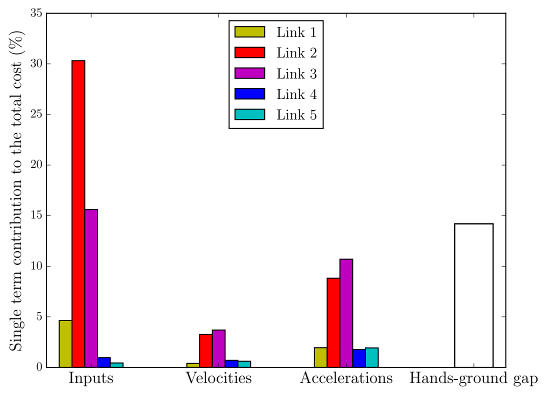


Figure 4.22: Percentages of the total cost related to each term of the objective function; “standing to supine” case.

Chapter 5

Solution Methodologies and Performance Comparisons

In Section 3.4 we illustrated the contact force models we developed, the first aim of this chapter is to analyze the effects that the contact stiffness and the tangential velocity reference have on the convergence process in order to identify suitable values for these parameters. After that different techniques to approach the problem presented in Chapter 4 are discussed. The third part of the chapter is focused on the effects that each cost term has on the behaviour of the humanoid, whereas at last a comparison between three alternative ways to deal with non-active contact forces (i.e. forces whose values are constantly equal to zero in a specific task) is performed.

5.1 Convergence Process Sensitivity to Contact Force Parameters

In this section we analyze the effects that the contact force parameters (introduced in Section 3.4) have on the performances of the optimization process, in particular we will focus our attentions on the contact stiffness κ and the tangential velocity reference $\dot{x}_{\text{ref}}$. Obviously an higher value of κ and a lower value of $\dot{x}_{\text{ref}}$ are generally to prefer but, as we will see, the use of excessively non-smooth functions leads to a loss of convergence velocity.

In order to have a deeper understanding of the effects that these values have on the efficiency of the optimization process, we conducted a set of tests. We considered a "supine to standing" task (Section 4.2.3) and, regarding the topics we will discuss in Section 5.2.1 and in Section 5.2.2, we took as example a planning where the optimization phase is split in a pseudo-static planning and a dynamic upgrade and contact forces are introduced as optimization variables. This task has been chosen because it leads to a motion that involves both sliding and steady phases of the hands-ground interaction.

In these tests we analyze the relation that links the number of iterations and the computation time to the values of the in question parameters. We decided to sample these quantities with an (almost) constant ratio scheme, starting from $\kappa = 2.0 \cdot 10^4 \ \frac{\text{N}}{\text{m}}$ and $\dot{x}_{\text{ref}} = 2.0 \cdot 10^{-2} \ \frac{\text{m}}{\text{s}}$ and multiplying each time these values by a factor ≈ 1.5. In Table 5.1 and in Table 5.2 we summarize the obtained results.

Stiffness κ $\left(\frac{\text{N}}{\text{m}}\right)$	**Iterations**	**Time** (s)
$2 \cdot 10^4$	167 (=145+22)	260 (=176+84)
$3 \cdot 10^4$	334 (=313+21)	388 (=315+73)
$5 \cdot 10^4$	286 (=265+21)	377 (=311+66)
$7 \cdot 10^4$	446 (=442+24)	498 (=425+73)
$1 \cdot 10^5$	2199 (=2015+184)	2138 (=1549+589)

Table 5.1: Effects on the convergence process of different values of the normal contact stiffness (a value of $\dot{x}_{\text{ref}} = 5.0 \cdot 10^{-2} \ \frac{\text{m}}{\text{s}}$ is considered).

Tangential velocity reference $\dot{x}_{\text{ref}}$ $\left(\frac{\text{m}}{\text{s}}\right)$	**Iterations**	**Time** (s)
$2 \cdot 10^{-2}$	loc. infeasibility	loc. infeasibility
$3 \cdot 10^{-2}$	562 (=527+35)	720 (=603+117)
$5 \cdot 10^{-2}$	286 (=265+21)	377 (=311+66)
$7 \cdot 10^{-2}$	232 (=212+20)	275 (=216+59)
$1 \cdot 10^{-1}$	298 (=279+19)	312 (=254+58)

Table 5.2: Effects on the convergence process of different values of the tangential velocity reference (a value of $\kappa = 5.0 \cdot 10^4 \ \frac{\text{N}}{\text{m}}$ is considered). The entry "loc. infeasibility" refers to the convergence in a point of local infeasibility of Ipopt interior-point algorithm.

As it can be seen from the tables, these parameters have have similar influence on the convergence process: excessively non-smooth values lead to a great increase of the computation time whereas an exaggeration in the opposite direction do not bring sufficient benefits to justify the loss of precision in the contact modeling. In both cases we decided to adopt compromise values: $\kappa = 5.0 \cdot 10^4 \ \frac{\text{N}}{\text{m}}$ and $\dot{x}_{\text{ref}} = 0.05 \ \frac{\text{m}}{\text{s}}$. The adopted value of stiffness has shown to require reasonably low running times and to conduce to admissible values of hands-ground penetration (in this case, the maximum distance evaluated during the contact phase was equal to 18 mm). In the case of the tangential velocity we settled for a value considerably lower than the characteristic sliding velocities of these tasks: in order to have an idea, in this example the velocity with which hands slide on the ground is equal to 0.55 $\frac{\mathrm{m}}{\mathrm{s}}$.

5.2 Alternative Approaches to the Problem Formulation

Optimal control problems can be can be defined in a great variety of alternative forms, for example some parameters can be expressed as variables of optimizations, contact forces can be defined as complementarity constraints, or optimization processes can be divided in series of problems of rising complexity. In this section the possibility to split the solution into two consecutive optimizations (introducing an intermediate problem where dynamic terms are neglected) and the effects of the definition of contact forces as optimization variables are discussed, a deepened analysis of the performances obtained by the exploitation of these methods is also proposed.

5.2.1 Pseudo-Static Intermediate Step

During the initial phases of the development of this project, we struggled even to obtain not very significant results; this was due to a combination of things: the great dynamic complexity of the in case problem, the use of less effective optimization algorithms (such as direct multiple shooting) and an imperfect knowledge of the CasADi framework. Because of these circumstances, we tried different strategies to solve the dynamic problem and one of them turned out to be very useful.

The application of Newton-type algorithms to non-convex problems leads to locally optimal solutions and, in this sense, initial guesses have a great influence on the nature of the movement we want to obtain. As pointed out in Section 4.1.4, the initial guess we use in this study is a linear interpolation and, as shown in Figure 4.2, this trajectory is quite different from an human-like motion. For this reason we thought that a split of the problem into two consecutive optimizations (introducing a pre-computation whose goal is to elaborate the linear interpolation in a more realistic trajectory) was a smart idea. Analyzing how this kind of motions are generally performed by humans it appear clear how accelerations and velocities are in general reduced to the minimum: since we have not a particular reason to do these actions very quickly, we prefer to give importance to the safety (stability) of the movement. In light of that we decided to run a first (fast and relatively easy to solve) problem, where the dynamic terms of the equations of motion are eliminated, and then upgrade the obtained solution with a second optimization where the whole dynamic equations are considered. Approaching the problem this way, most of the work is done during the first phase (reducing the overall computation time) while the upgrade, in general, needs a few iterations.

To ensure that the obtained initial guess can be useful, we have to set a cost function that drives the solution to a quasi-static behaviour; in particular a penalization of the velocities and the accelerations is needed (as exposed in Section 4.1.2). In this case, in fact, we can predict that the approximation $B(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) - Q(q,\dot{q},u) \approx G(q) - Q(q,\dot{q},u)$ will hold. It's interesting also to note that a strong penalization of accelerations and velocities leads to a great reduction of the possibilities to move from a configuration to the other; thinking for example at the case of the interpolation, in absence of accelerations (and remembering the ankle constraint Equation 3.2) the only way to complete the task is by using the hands to bring the center of mass above the feet, in order to be able to use knees and hips to stand up. This restriction of the set of feasible solutions has pros and cons: while on the one hand this guides the solution to have specific characteristics, on the other it might create narrow connections between subsets of $\mathcal{C}$ that might be hard to find by the solver.

As it can be seen from the previous relation, because of $\dot{q}$ even the simplified expression has a dependence from time (due to the approximation of the Coulomb model with an hyperbolic function of the velocity). This approximation has to be maintained: if we remove this relation the robot will have to move over a frictionless ground, whereas we want a motion that takes advantage of the friction. For this reason this approach to the problem cannot properly be called static and, in this sense, we coined the term "pseudo-static". Anyway this impossibility of removing time from the simplified expressions also have a positive side: looking at the discrete time problem, this approximation does not remove any variable or equation (note that the kinematic reconstruction is still needed to relate positions and velocities of different steps); this means that the problem does not change structure and the same cost functions and the same optimization variables can be used.

Summarizing, we split our planning in the solution of two consecutive problems whose characteristics are illustrated in Table 5.3. Section 5.2.3 concerns with a numerical investigation of the benefits of this technique and the one presented in Section 5.2.2.

5.2.2 Different Methods to Introduce Contact Force Constitutive Relations

Given the functions that relate tangent and normal contact forces to positions and velocities, we interrogate ourselves whether it is better to introduce contact force models

Settings	**Pseudo-static solution**	**Dynamic solution**
Initial guess	linear interpolation	pseudo-static solution
Cost function	$L_u + L_{\dot{q}} + L_{\ddot{q}} + L_{y_H}$	$L_u + L_{\dot{q}} + L_{\ddot{q}} + L_{y_H}$
DT ODE model	$G_{i+\frac{1}{2}} = Q_{i+\frac{1}{2}}$ $q_{i+1} - q_i = \dot{q}_{i+\frac{1}{2}}\Delta t$	$B_{i+\frac{1}{2}}\frac{\dot{q}_{i+1}-\dot{q}_i}{\Delta t} + C_{i+\frac{1}{2}}\dot{q}_{i+\frac{1}{2}} + G_{i+\frac{1}{2}} = Q_{i+\frac{1}{2}}$ $q_{i+1} - q_i = \dot{q}_{i+\frac{1}{2}}\Delta t$

Table 5.3: Partition of the problem in two consecutive plannings. The notation must be interpreted as follows: i represents the generic step, $B_{i+\frac{1}{2}} = B(q_{i+\frac{1}{2}})$, $C_{i+\frac{1}{2}} = C(q_{i+\frac{1}{2}}, \dot{q}_{i+\frac{1}{2}})$, $G_{i+\frac{1}{2}} = G(q_{i+\frac{1}{2}})$, $Q_{i+\frac{1}{2}} = Q(q_{i+\frac{1}{2}}, \dot{q}_{i+\frac{1}{2}}, u_i)$, $q_{i+\frac{1}{2}} = \frac{q_i+q_{i+1}}{2}$, and $\dot{q}_{i+\frac{1}{2}} = \frac{\dot{q}_i+\dot{q}_{i+1}}{2}$.

directly in the computation of the equations of motion (embedded approach) or to define extra variables that, in the later optimization phase, are forced to be equal to their expression by an equality constraint (augmented approach).

We can try to foresee the results analyzing how these two different approaches act on the iterations of the optimal control problem. To do this we introduce the following scalar example

$$m(x)\ddot{x} + c(x,\dot{x})\dot{x} = u + F(x,\dot{x}),$$

where F represent a generic external force that is a function of the position and the velocity. This equation of motion can be discretized, for simplicity, as

$$\begin{aligned}
&m(x_i)\frac{\dot{x}_i - \dot{x}_{i-1}}{\Delta t} + c(x_i,\dot{x}_i)\dot{x}_i = u_{i-1} + F(x_i,\dot{x}_i),\\
&x_i - x_{i-1} = \dot{x}_i\Delta t.
\end{aligned}$$

Without considering the fact that using Ipopt we have to introduce constraints as inequalities, we can analyze what are the effects that these two different methods of dealing with contact forces have on the first order optimality conditions (presented in Appendix A)

$$\nabla\phi(v) + \nabla\chi(v)\lambda = 0,$$

where v is the optimization variable, $\phi(v)$ is the cost function and $\chi(v)$ are the equality constraints. In other words: what is the most effective way to make the solver aware that contact forces can be exploited to reach goal configuration and to reduce input torques?

Embedded Definition of Contact Force Models

In this case our variables of optimization are x_i, $\dot{x}_i$ and u_{i-1}, while all the equality constraints where these variables appear are

$$\chi_i(v) = \begin{bmatrix} m(x_i)\frac{\dot{x}_i-\dot{x}_{i-1}}{\Delta t} + c(x_i,\dot{x}_i)\dot{x}_i - u_{i-1} - F(x_i,\dot{x}_i) \\ x_i - x_{i-1} - \dot{x}_i\Delta t \\ m(x_{i+1})\frac{\dot{x}_{i+1}-\dot{x}_i}{\Delta t} + c(x_{i+1},\dot{x}_{i+1})\dot{x}_{i+1} - u_i - F(x_{i+1},\dot{x}_{i+1}) \\ x_{i+1} - x_i - \dot{x}_{i+1}\Delta t \end{bmatrix} = 0,$$

where the notation $\chi_i(v)$ isn't quite proper (since in this case the index i isn't define in the same set as the original one). Remembering that the Lagrange multiplier vector has the dimension of the vector of the equalities, in this case we have (with the same uncorrect notation) $\lambda_i \in \mathbb{R}^4$.

During the optimization process, the possibility of interacting with the environment lies in the terms of $\nabla\chi(v)$ where the dynamic constraints are differentiated with respect to positions or velocities. The terms of this gradient matrix have the following form:

$$\nabla\chi_i(v) = \begin{bmatrix} \frac{\partial m(x_i)}{\partial x_i}\frac{\dot{x}_i-\dot{x}_{i-1}}{\Delta t} + \frac{\partial c(x_i,\dot{x}_i)}{\partial x_i}\dot{x}_i - \frac{\partial F(x_i,\dot{x}_i)}{\partial x_i} & 1 & 0 & -1 \\ \frac{m(x_i)}{\Delta t} + \frac{\partial c(x_i,\dot{x}_i)}{\partial \dot{x}_i}\dot{x}_i + c(x_i,\dot{x}_i) - \frac{\partial F(x_i,\dot{x}_i)}{\partial \dot{x}_i} & -\Delta t & -\frac{m(x_{i+1})}{\Delta t} & 0 \\ -1 & 0 & 0 & 0 \end{bmatrix}.$$

Looking at the gradient matrix it is possible to see how the influence of the contact forces lies exclusively in the $[1:2,1]$ terms, where their expression is differentiated. Anyway there is not any guarantee that such terms are big enough to prevail on the remaining derivatives of the dynamic constraints and that they are able to drive the optimization in the direction where the contact forces increase. In order to have some numbers as reference for the humanoid case, we report as example that using the values assigned in Section 3.4 we obtain

$$\left.\frac{\partial F_n}{\partial y}\right|_{y=20\text{ cm}} \approx -10^{-3}\ \frac{\text{N}}{\text{m}},\quad \left.\frac{\partial F_n}{\partial y}\right|_{y=50\text{ cm}} \approx -10^{-14}\ \frac{\text{N}}{\text{m}}.$$

As it can be seen, because of the extremely high value of contact stiffness, these gradients tend to zero very rapidly and we can expect that in general (with the hands separated from the ground) they are negligible with respect to the others members of the derivatives.

To conclude we can state that using this method we are certain that, at every iteration, contact forces will coincide with their expressions (that is not necessary a positive aspect) and that, if the hands are separated from the ground, the derivatives of these functions may become too small to drive the optimization to a hands-ground interaction.

Augmented Definition of Contact Force Models

An alternative to the direct introduction of the contact forces model in the computation of the external force vector, is the definition of additional optimization variables and the introduction of a set of equality constraints that relate these new variables with the positions and the velocities of their points of application. Doing this we are adding to the original $2 \cdot n \cdot (N+1) + n \cdot N = 1510$ other $2 \cdot n \cdot N = 1000$ optimization variables and, in this sense computation time per step will surely be longer (properly speaking the number of variables we introduce is in general ≤ 1000 since sometimes we do not introduce all the contact forces in the equations of motion).

Denoting these additional optimization variable at the generic step i as F_i, the vector that contains all the equalities where our generic variables (x_i, $\dot{x}_i$, u_{i-1}, and F_i) are present assume the form

$$\chi_i(v) = \begin{bmatrix} m(x_i)\frac{\dot{x}_i - \dot{x}_{i-1}}{\Delta t} + c(x_i, \dot{x}_i)\dot{x}_i - u_{i-1} - F_i \\ x_i - x_{i-1} - \dot{x}_i \Delta t \\ F_i - F(x_i, \dot{x}_i) \\ m(x_{i+1})\frac{\dot{x}_{i+1} - \dot{x}_i}{\Delta t} + c(x_{i+1}, \dot{x}_{i+1})\dot{x}_{i+1} - u_i - F_{i+1} \\ x_{i+1} - x_i - \dot{x}_{i+1}\Delta t \\ F_{i+1} - F(x_{i+1}, \dot{x}_{i+1}) \end{bmatrix} = 0.$$

Remembering once again that the number of Lagrange multipliers is equal to the number of equality constraints, in this case we have $\lambda_i \in \mathbb{R}^6$.

This time the gradient matrix of the equality constraints is

$$\nabla\chi_i(v) = \begin{bmatrix} \frac{\partial m(x_i)}{\partial x_i}\frac{\dot{x}_i - \dot{x}_{i-1}}{\Delta t} + \frac{\partial c(x_i,\dot{x}_i)}{\partial x_i}\dot{x}_i & 1 & -\frac{\partial F(x_i,\dot{x}_i)}{\partial x_i} & 0 & -1 & 0 \\ \frac{m(x_i)}{\Delta t} + \frac{\partial c(x_i,\dot{x}_i)}{\partial \dot{x}_i}\dot{x}_i + c(x_i,\dot{x}_i) & -\Delta t & -\frac{\partial F(x_i,\dot{x}_i)}{\partial \dot{x}_i} & -\frac{m(x_{i+1})}{\Delta t} & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

In this case when the derivatives of the dynamic constraints are performed, forces aren't differentiated unless the variable of differentiation are the forces themselves (term $[4, 1]$), whereas the derivatives of the forces expressions (with respect to positions or velocities) are executed exclusively when the additional constraints are differentiated (terms $[1:2, 3]$). If we compare this matrix with the one of the previous section, we can see how in this case the derivatives of the contact forces expressions are not added to the remaining dynamic terms and, moreover, they are multiplied to new dedicated Lagrange multipliers. In light of this we can say that, during optimization, contact forces derivatives are isolated from the remaining dynamic constraints and these new multipliers might be used to first achieve more relevant restrictions (such as the minimization of the inputs

		Embedded		**Augmented**	
		$\mathcal{D}$	$\mathcal{PS}$+$\mathcal{D}$	$\mathcal{D}$	$\mathcal{PS}$+$\mathcal{D}$
$\mathfrak{T}_1$	iterations	200	420 (=383+37)	181	323 (=291+32)
	time (s)	311	386 (=337+49)	707	555 (=442+113)
$\mathfrak{T}_2$	iterations	317	723 (=703+20)	193	225 (=209+16)
	time (s)	648	467 (=440+27)	827	359 (=308+51)
$\mathfrak{T}_3$	iterations	953	1133 (=1114+19)	346	286 (=265+21)
	time (s)	1408	618 (=592+26)	1170	377 (=311+66)
$\mathfrak{T}_4$	iterations	86	283 (=258+25)	91	269 (=250+19)
	time (s)	128	210 (=174+36)	257	305 (=240+65)

Table 5.4: Comparison between the embedded and the augmented definition of contact force models and between the direct dynamic solution ($\mathcal{D}$) and the two steps optimization with the computation of a pseudo-static initial guess ($\mathcal{PS}$+$\mathcal{D}$). $\mathfrak{T}_1$ represents the "prone to standing" task, $\mathfrak{T}_2$ the "standing to prone" task, $\mathfrak{T}_3$ the "supine to standing" task, and $\mathfrak{T}_4$ the "standing to supine" task.

or the achievement of the final configuration) than the equalities between contact force variables and their expression.

5.2.3 Numerical Validation

In this section we analyze from a numerical point of view the different techniques proposed in Section 5.2.1 and in Section 5.2.2: on the one hand we have the presumed increase of performances caused by a division of the problem into two consecutive optimizations; on the other hand we have the differences between an embedded definitions of the contact force constitutive equations and an augmented formulation of the NLP, where these forces are introduced as optimization variables. To compare these alternatives we performed a series of tests where all the four elementary tasks (exposed in Section 4.2) are investigated; each one of these tasks has been approached in four different ways (originated by the combinations of the 2+2 previous alternatives). In particular we focused our attention on the number of iterations and the total time necessary to obtain each solution and, in order to find possible discrepancies between the computed trajectories, a check is also performed. In Table 5.4 all the results are reported.

We start by saying that all the alternatives proposed in this section gave identical results in term of computed trajectory, except for the "supine to standing" task where, because of the low weight related to the inputs penalization in the cost function, the intermediate step drove the solution to assume a different behaviour.

Analyzing the advantages caused by the introduction of a pseudo-static step, it is interesting to note that the number of iterations necessary to find the intermediate results are generally higher than the ones necessary to obtain a direct dynamic solution, fact that is in contrast with our reflection on the ease of solution of the pseudo-static optimization. Anyway in the pseudo-static case each iteration requires almost half of the time (than a dynamic one) and an overall reduction of computation time can be seen (especially in case of a "supine to standing" task). As it can be seen in all the cases the dynamic upgrade to the pseudo-static solution required very short amount of iterations and time. We also underline how the pseudo-static approach leads to an general regularization of the computation times, while the direct dynamic approach shows great variations of time from a task to the other.

For what concerns the comparison between the embedded and the augmented definition of contact forces, a general reduction of the number of iterations is caused by the definition of contact forces as optimization variables but in this case each iteration needs approximately twice the time. The addition of these two aspects leads to similar performances of these methods, with a slight greater effectiveness of the embedded technique.

In conclusion we can state that the intermediate step is helpful in terms of computation time but it is not essential, whereas the excessive increase of the time per step ratio of the augmented approach gets the better of its higher efficiency.

5.3 Taking Non-Active Contact Forces into Account

As pointed out before, the codes we developed are equipped with the option of limiting the number of the considered contact forces to the minimum necessary. This operation is acted with the goal of reducing as possible the complexity of the dynamic equations, with the consequent reduction of the computation time. Furthermore in case of a symbolic representation of the contact forces, this avoids the introduction of hundreds of optimization variables. On the other hand, this fact can be seen as a limitation to the possible interactions between the robot and the environment. For this reason we decided to conduct a test which goal is to compare the nature of the trajectory and the performances of different ways to take non-active contact forces into account. As a reference the solution of the "standing to supine" case (presented in Section 4.2.3) is considered, and two additional alternative approaches are investigated. In the first case we simply analyze the results obtained by introducing all the contact forces ($A_{\text{cf}} \equiv \{1, \ldots, n\}$). In the second case we add a series of geometric constraints that avoid the contact with the ground of

		Embedded		**Augmented**	
		$\mathcal{D}$	$\mathcal{PS}$+$\mathcal{D}$	$\mathcal{D}$	$\mathcal{PS}$+$\mathcal{D}$
$\mathcal{M}_1$	iterations	953	1133 (=1114+19)	346	286 (=265+21)
	time (s)	1408	618 (=592+26)	1170	377 (=311+66)
$\mathcal{M}_2$	iterations	676	1391 (=1372+19)	355	268 (=247+21)
	time (s)	1120	1342 (=1315+27)	2363	489 (=357+132)
$\mathcal{M}_3$	iterations	239	1233 (=1214+19)	379	255 (=234+21)
	time (s)	283	489 (=468+21)	775	398 (=363+35)

Table 5.5: Comparison between the performances obtained by the introduction of: the minimum necessary number of contact forces ($\mathcal{M}_1$), the total number of contact forces ($\mathcal{M}_2$), and the minimum number of contact forces with additional geometric constraints that check possible collisions of non-active joints ($\mathcal{M}_2$). The entry "$\mathcal{D}$" refers to the direct dynamical approach, while "$\mathcal{PS}$+$\mathcal{D}$" is related to the double step solution of the dynamic problem (with a pseudo-static intermediate optimization).

the excluded joints ($y_{O_i} > 0$ with $i \in \{1, \ldots, n\} \backslash A_{\text{cf}}$).

As exposed in Section 4.2.3, for the in case problem the contact forces that are necessary for the movement are the ones applied to the hips and the hands. In light of this, the first alternative proposed consists in the introduction of three additional contact forces (i.e. $2 \cdot 3 \cdot N = 600$ extra optimization variables) while the second method leads to a collision detection of three joints. In Table 5.5 the obtained results are summarized.

We start by reminding that, as pointed out in Section 5.2.3, for the in case task the trajectory obtained with the double step approach is slightly different form the one given by the direct solution of the dynamic problem. Using these alternative techniques to deal with non-active forces we observed the same pattern: all the double step optimizations computed the same trajectory and all the direct dynamic problems led to the same solution, except for the collision detection ($\mathcal{M}_3$) embedded dynamic ($\mathcal{D}$) approach that brought to a movement equal to the one obtained using the double step method.

Comparing the first two alternatives (minimum and total number of contact forces introduced), the first aspect to underline is that the number of iterations of these two approaches are similar, this fact suggests that the introduction of additional contact forces do not have the tendency to drive the solver to different solutions. The increased number of variables is reflected in a general loss of performance in terms of computation time.

Going on to the analysis of the effects of the collision detection of the non-active joints, astonishing results are obtained: this approach leads to a general reduction of the computation time even if compared with the first technique where non-active joints are not monitored any way! In some cases a decrease of the time per iteration ratio is also registered.

To conclude we can state that the introduction of geometric constraints to avoid collisions between non-active joints and the ground is certainly the best way to ensure that the computed trajectories are actually exploitable.

5.4 Single Cost Term Influence on the Resulting Trajectories

In Section 4.1.2 we illustrated what are the terms that compose our objective function, we are now interested in deepening our comprehension of the effects that each one of these terms has on the final behaviour of the humanoid. To do that we analyze the usual “supine to standing” task (presented in Section 4.2.3) and we perform a series of tests where, each time, we reduce by a factor of 10^3 all but one of the cost function weights W. The reason why we only reduce the terms that aren’t object of the in case investigation is that their presence, even with a minimal influence, avoids completely unnatural movements (like the one that we could obtain with a penalization of the hands-ground distance only).

In this case we preferred to directly run a dynamic optimization (referring to the alternatives presented in Section 5.2.1), this choice is mainly due to the fact that a reduction of the weight related to the accelerations makes a pseudo-static result unsuitable as initial guess. Furthermore a direct dynamic optimization does not influence in any way the final behaviour of the humanoid. For what concerns the other techniques we presented, in this case we exploited an augmented formulation of contact forces whereas we turned off non-active forces. In order to have a comparison, each one of the following figures includes in the background a red profile that represents the original movement (obtained with the simultaneous use of every cost term).

Starting from the effects that the penalization of the input torques has on the resulting movement, in Figure 5.1 we propose the trajectory obtained reducing all but one the cost terms presented in Section 4.2.3. As it can be seen, in this case the main tendency of the robot is to place itself in a vertical configuration that ideally eliminate the necessity of any input. This position is achieved with an intense exploitation of the arms inertia (since the related weight is relatively low): as shown in Figure 5.2 the humanoid support the rising phase with impulsive torques applied by the shoulder and the elbow actuators. Figure 5.2 also report the percentages of the total cost to ascribe to each term; as it can be seen, despite of its reduced value, the weight related to the accelerations avoids an excessively quick movement from the initial position to the vertical one. The amount of time necessary to the computation of this result is 946 s whereas the number of iteration is equal to 245.

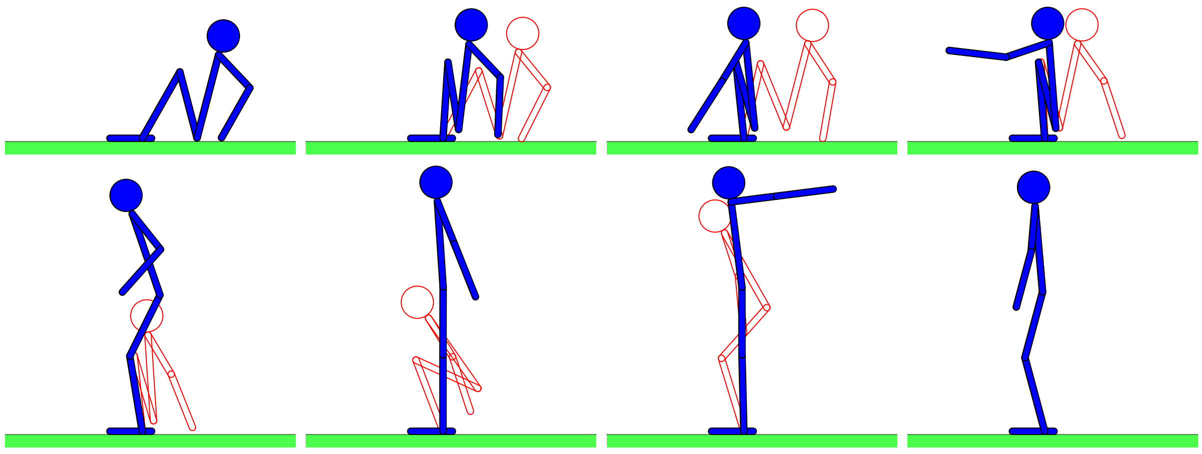


Figure 5.1: Effects of a strong penalization of the inputs torques on the "supine to standing" motion (blue), compared with the reference result (red); 0.71 s per frame.

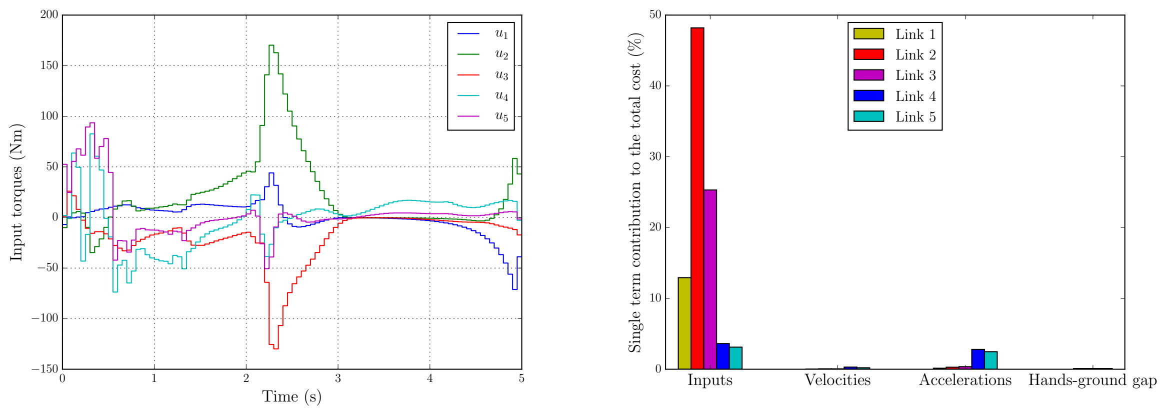


Figure 5.2: Input torques as a function of time and percentages of the total cost related to each term of the objective function; "standing to supine" case with a strong penalization of the input torques.

Moving on to the analysis of the effects of the term that penalize the velocities, in Figure 5.3 the obtained trajectory is illustrated. As logical, in the first phase of the movement the humanoid strongly accelerates in order to reach velocities that are maintained almost constant for the remaining time. Note that this trajectory lead to great effort of the knee actuator. Figure 5.4 confirms the previous observations and shows how, this time, only a weak connection between the penalization of the velocities and the inputs is present. This time the optimization required 694 s and 199 iterations.

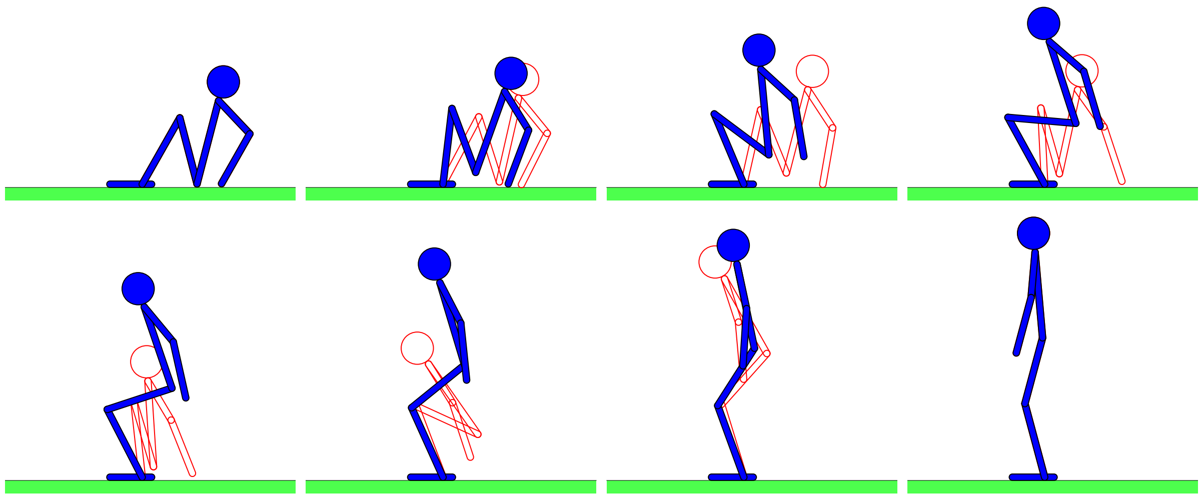


Figure 5.3: Effects of a strong penalization of the velocities on the “supine to standing” motion (blue), compared with the reference result (red); 0.71 s per frame.

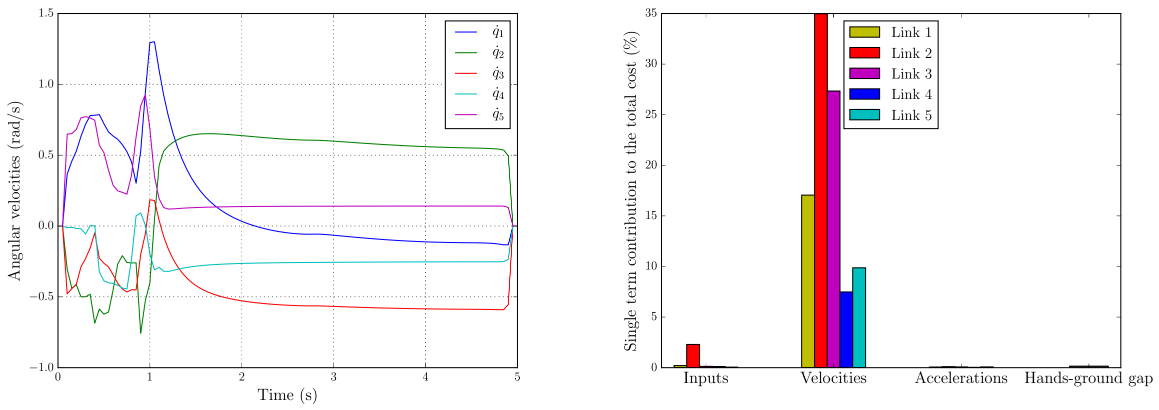


Figure 5.4: Angular velocities as a function of time and percentages of the total cost related to each term of the objective function; “standing to supine” case with a strong penalization of the angular velocities.

In Figure 5.5 the trajectories obtained by a high penalization of the accelerations are illustrated. This time the computed motion (blue) is quite similar to the original one (red), the main difference is shown in the central phase of the movement where the reference trajectory is strongly influenced by the penalization of the input torques and, for this reason, the humanoid assumes a curled configuration. In Figure 5.6 the accelerations as a function of time for the in case result are proposed. Figure 5.6 also represent the percentages of the total cost to ascribe to each term, in this case a weak relation with the cost terms associated to the inputs and the hands-ground distance can be observed. In this case the running time was equal to 883 s and 246 iterations were needed.

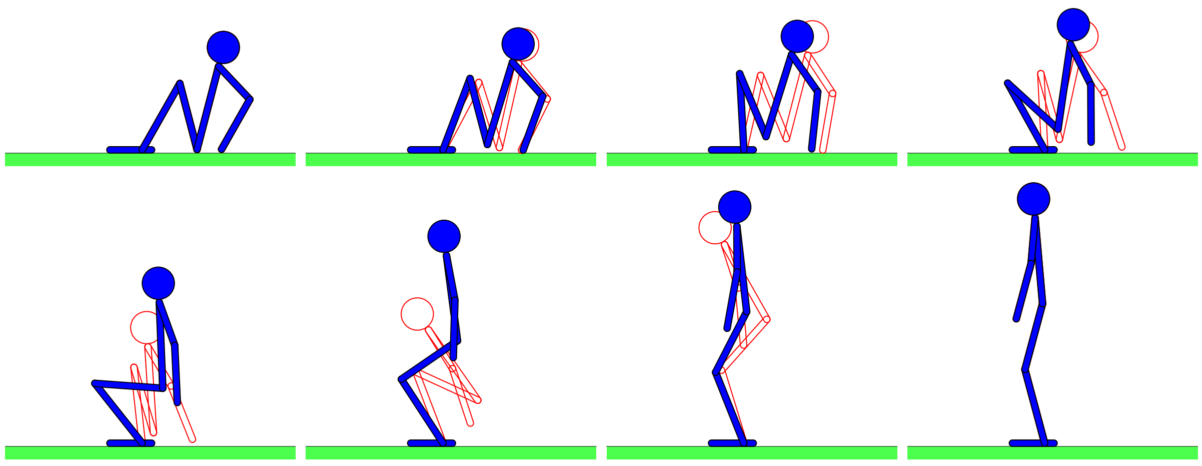


Figure 5.5: Effects of a strong penalization of the accelerations on the “supine to standing” motion (blue), compared with the reference result (red); 0.71 s per frame.

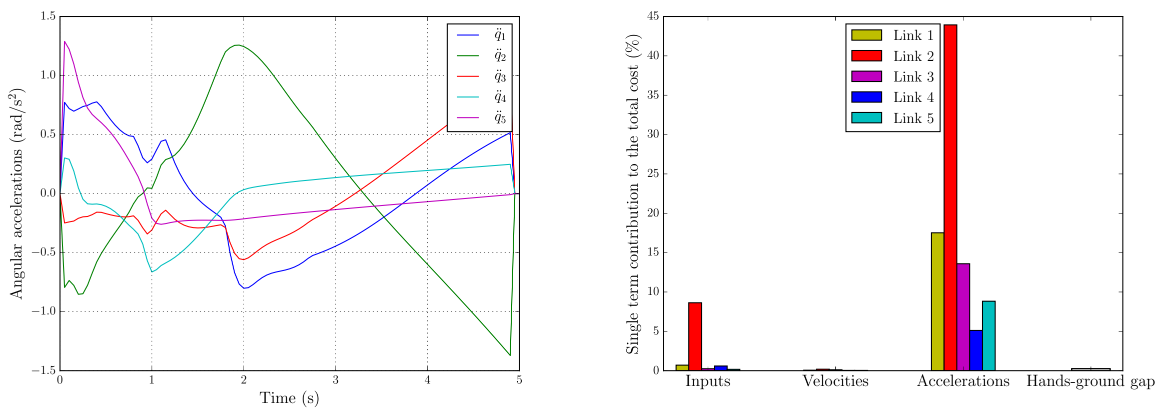


Figure 5.6: Angular accelerations as a function of time and percentages of the total cost related to each term of the objective function; “standing to supine” case with a strong penalization of the angular accelerations.

Finally we report the results obtained imposing to the humanoid an inversely proportional to the time penalization of the hands-ground distance. As it can be seen this motion (presented in Figure 5.7) consists in a rise of the center of mass of the system while the hands remain as long as possible in contact with the ground and all the stand-up movement is concentrated in the last 0.7 s, where the humanoid makes a very quick movement to reach the goal position. Figure 5.8 shows how this time an high penalization of a single term is unavoidably in conflict with the effects caused by the other terms of the objective function. In particular the tendency to leave the hands as long as possible in contact with the ground leads to high values of velocities and accelerations in the very last part of the movement. Figure 5.8 also illustrate the evolution of the contact forces applied by the ground to the hands; as it can be seen, the original aim of this term is achieved: since the hands have to be in contact with the floor, the humanoid has the tendency to load as possible this contact in order to reduce to the minimum the input torques. The time necessary to plan this motion was equal to 1272 s whereas 316 iterations were needed.

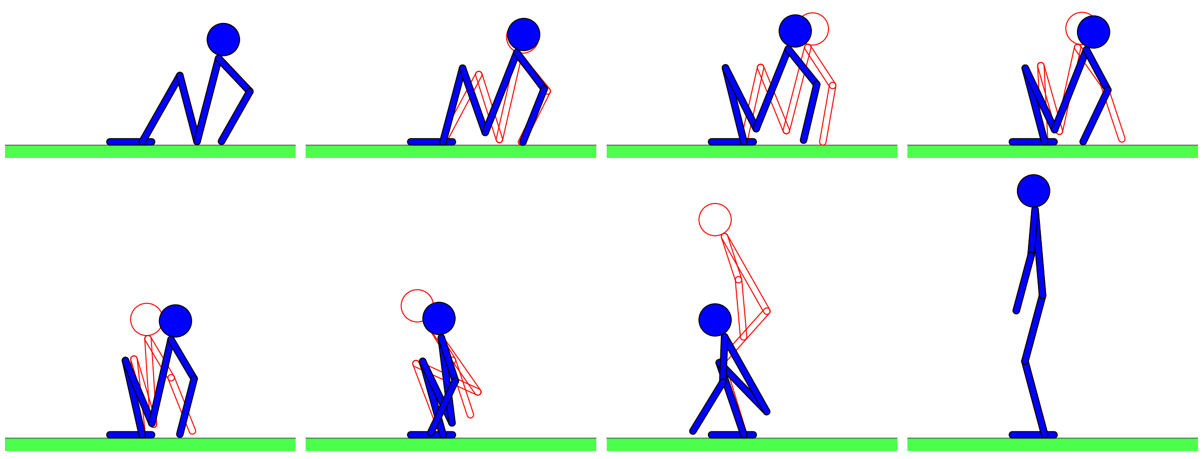


Figure 5.7: Effects of a strong penalization of the hands-ground distance on the "supine to standing" motion (blue), compared with the reference result (red); 0.71 s per frame.

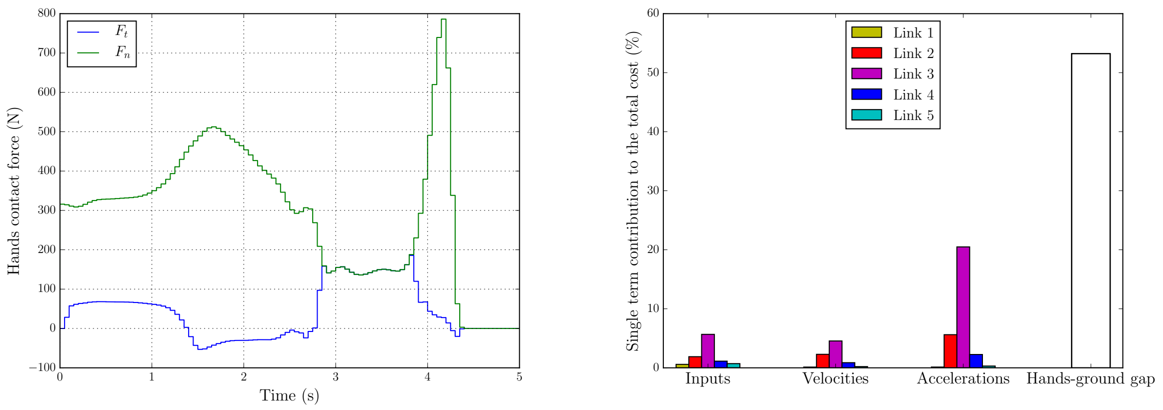


Figure 5.8: Normal and tangential hands contact forces as a function of time and percentages of the total cost related to each term of the objective function; "standing to supine" case with a strong penalization of the hands-ground distance.

Chapter 6

Further Improvements

In this study we demonstrated the possibility to exploit numerical optimization methods for motion planning of complex robotic systems. We hope that the presented results will encourage future works to deepen the pointed out aspects. On our part, there are many topics we want to analyze and develop; in this section we briefly summarize our future aims.

A deeper knowledge of the frameworks and the used algorithms is probably the first step to make. During this study we realized how a better organization of the codes might have a great influence on the computation times and even on the nature of results. There is a wide range of slightly different possibilities to approach these problems, and sometimes choose one instead of another can make a failure become a success. Furthermore, the use of high-level interfaces lead to a great capability of quickly implementing very complex and efficient algorithms but, on the other hand, it precludes the possibility to understand the real complexity of the tools we are exploiting.

From a physical point of view, an in-depth analysis about the different alternatives to write dynamic equations has to be conducted. As exposed in Chapter 3 the expression of some of the geometric and dynamic entities in a closed form gave us the opportunity to drastically reduce the number of nodes of the dynamic equations; even if this method worked in our case, this is an approach tailored for the in case problem and it cannot be extended to more complex or three-dimensional problems. A pragmatic comparison between the form and the complexity of the expressions generated by methods of different nature (such as Euler-Lagrange and Newton-Euler approaches) has to be made. Even some variations to the standards application of the Newton-Euler equation might give some interesting results, for example the various alternatives to choose the location of the pivot point of the rotational equilibria can result in the systematic elimination of some undesired terms. In Section 3.7 we pointed out the possibility to exploit the structure of the optimization problem to develop new approaches to the definition of the dynamic constraints, in this case taking advantage from the efficiency with which these algorithms treat equality constraints we tried to reduce to the minimum the complexity of the dynamics and to burden as possible the optimization routine. Moreover it is fundamental to remember that the goal of all these analysis is to reduce the computation time (that is strictly related to number of nodes of the equations) and, as illustrated in Section 3.1, sometimes even very trivial tricks can make the difference in this sense.

In Section 4.1.2 we introduced the basic concepts of multi-objective optimization; going on with the application of numerical optimization techniques this is a subject that must be considered. Bad-scaled problems and inadequate cost functions make the relation between the results of the motion planning problem and the inputs (such as the weights of the cost functions) discontinuous. As pointed out, in case of a non-convex Pareto front the use of a weighted sums of the cost terms leads to the impossibility to explore all the optimal solutions; this fact is unacceptable considering that the main goal of motion planning is exactly the exploration of the different solutions to the problem.

Using the technique presented in Section 5.2.1 some other topics came to our attentions. Numerical optimization methods are extremely sensitive to the use of adequate initial guesses. A basic approach to this problem consists in the introduction of only the initial values of the primal optimization variables, not remembering that also dual variables and barrier function parameters are objects of the converge process. For example when a pseudo-static solution is used to warm start the dynamic problem, the optimization process begins its search without the knowledge of the values of the Lagrange multipliers and with extremely relaxed barrier parameters. In this sense, the introduction of complete initial guesses might greatly speed up the convergence of the second optimization.

Another field where we lack of knowledge is the one of the solvers. Ipopt gives a wide selection of linear solvers for sparse problems, in this study we used the ma57 routine but, with some superficial tries, we realized that in some cases there might be better choices.

Once the planning problems will be adequately analyzed, the natural evolution of this approach consists in the *Nonlinear Model Predictive Control* (NMPC) technique. This method exploits the structures presented in this study, to on-line compute new optimal trajectories in case the measured configurations are different from the ones established. If each planning is fast enough to be computed in real time, NMPC gives the ability to control the robot with great effectiveness.

Finally we have to remember how the original aim of the thesis was the development of hybrid approaches to the motion planning problem. In Section 1.1.2 we briefly overview the world of the sampling-based algorithms but these are only one of many. Recently some others techniques (for example the ones based on the Lyapunov stability theory) has been proposed, leading the way to a new set of prospective advancements.

Appendix A

Elements of Numerical Optimization

A positive aspect of using optimization software (such as Ipopt) is that they make you able to deal with complex optimization problems very quickly, even if your mathematical background is not as wide as it should be. On the other hand, non-convex optimization problems present a great variety of aspects that needs to be in depth understood in order to obtain valid and generalizable results. Because of that, in this appendix, we introduce, from a very basic point of view, some fundamental concepts of non-convex optimization. This analysis doesn't want to be an exhaustive dissertation on numerical optimization, only the topics related to the methods we've used during our studies will be analyzed; furthermore we underline that practical optimization algorithms present much more complicated structures than the ones presented here (for a complete and very detailed analysis see [19]). The presentation of these topics is structured following [2].

In the first section essential concepts of nonlinear optimization are introduced. In the second part we will present *Interior-Point Method* (IPM), an efficient algorithm to solve inequality constrained optimization problems. The third section treats *Algorithmic Differentiation* (AD), an accurate and very efficient routine to compute derivatives.

Tools we present in this appendix will give us a clearer idea about how NLPs derived in Chapter 2 can be handled.

A.1 Optimality Conditions

In this section we illustrate the first and second order optimality conditions, that represent a set of necessary and sufficient conditions to state that a point is a local minimizer of a given function (subject to both equality and inequality constraints).

As seen in Chapter 2 direct methods lead to the solution of NLPs with the following general form.

Problem 1 (Non-Linear Program)**.**

$$\begin{array}{lll} \underset{v}{\textit{minimize}} & \phi(v) & \textit{(objective function)} \\ \textit{subject to} & \chi(v) = 0 & \textit{(equality constraints)} \\ & \psi(v) \leq 0 & \textit{(inequality constraints)} \end{array}$$

where $\phi : \mathbb{R}^{n_v} \to \mathbb{R}$, $\chi : \mathbb{R}^{n_v} \to \mathbb{R}^{n_\chi}$ and $\psi : \mathbb{R}^{n_v} \to \mathbb{R}^{n_\psi}$ are assumed to be twice continuously differentiable.

Let $\Omega := \{v \in \mathbb{R}^{n_v} \mid \chi(v) = 0, \psi(v) \leq 0\}$ denote the *feasible set* (set of all points that satisfy the constraints) and $v^* \in \Omega$ be a *feasible point.* A feasible point needs to satisfy two conditions to be called a solution of the NLP; likewise unconstrained optimization problems need a first order condition ($\nabla\phi(v^*) = 0$) and a second order condition ($\nabla^2\phi(v^*) \succ 0$).

Before we can define the optimality conditions, we need to describe the feasible set in the neighborhood of the candidate point v^*; it may be that not all the inequality constraints need to be considered. We define the i^{th} inequality constraint *active* if and only if $\psi_i(v^*) = 0$ and *inactive* otherwise. Let us also denominate the index set of all active inequality constraints $\mathcal{A}(v^*) \subset \{1, 2, ..., n_\psi\}$ as the *active set.*

With the purpose of verifying if v^* is a local minimizer, we have to check the following qualification.

Definition 1 (Linear Independence Constraint Qualification)**.** *The LICQ holds at $v^* \in \Omega$ iff all vectors $\nabla\chi_i(v^*)$ for $i \in \{1, 2, ..., n_\chi\}$ and $\nabla\psi_i(v^*)$ for $i \in \mathcal{A}(v^*)$ are linearly independent.*

Defining a vector where we stack all the equality constraints and the active inequality constraints

$$\tilde{\chi}(v) = \begin{bmatrix} \chi(v) \\ \psi_i(v),\ i \in \mathcal{A}(v^*) \end{bmatrix} : \mathbb{R}^{n_v} \to \mathbb{R}^{n_{\tilde{\chi}}}$$

the LICQ is satisfied if the Jacobian $\frac{\partial\tilde{\chi}}{\partial v}(v^*)$ has full row rank.

For inequality constrained problems the equivalents of the first order conditions of optimality are the *Karush-Kuhn-Tucker optimality conditions.* Generally speaking (for non-convex problems) these are only necessary conditions to state that v^* is a local minimizer but, in the instance of a convex problem, they are sufficient for global optimality.

Theorem 1 (Karush-Kuhn-Tucker optimality conditions)**.** *If v^* is a local minimizer of the NLP and the LICQ holds at v^*, there exist Lagrange multipliers vectors $\lambda^* \in \mathbb{R}^{n_\chi}$ and*

$\mu^* \in \mathbb{R}^{n_\psi}$ such that

$$\begin{aligned}
&\nabla_v \mathcal{L}(v^*, \lambda^*, \mu^*) = 0,\\
&\chi(v^*) = 0,\\
&\psi(v^*) \leq 0,\\
&\mu^* \geq 0,\\
&\mu^{*T}\psi(v^*) = 0,
\end{aligned}$$

where $\mathcal{L}(v, \lambda, \mu) = \phi(v) + \lambda^T\chi(v) + \mu^T\psi(v)$ is the Lagrangian function whereas a triple (v^*, λ^*, μ^*) that satisfies LICQ and KKT conditions is called KKT point.

The last three conditions exposed in the KKT problem are called *complementarity conditions* and for each index $i \in \{1, 2, ..., n_\psi\}$ they define an L-shaped set in the $(\psi_i(v), \mu_i)$ space. Each one of these sets is a non-smooth manifold with a non-differentiable point in the origin. We call *weakly active* a constraint for which our candidate v^* gives $\psi_i(v^*) = 0 \ \vee \ \mu_i^* = 0$ and *strictly active* a constraint for which our candidate v^* gives $\psi_i(v^*) = 0 \ \vee \ \mu_i^* > 0$. Eventually we say that *strict complementarity* holds if and only if, for a specific KKT point, all active constraints are strictly active.

In case of strict complementarity of a KKT point, the optimization problem can (locally) be seen as a problem with equality constraints only (the ones defined by the vector $\tilde{\chi}(v)$).

At this point we are able to introduce the *Second Order Optimality Conditions.*

Theorem 2 (Second Order Optimality Conditions)**.** *Consider: a point v^* where LICQ holds, multipliers λ^* and μ^* such that the KKT conditions are satisfied, a basis $Z \in \mathbb{R}^{n_v \times (n_v - n_{\tilde{\chi}})}$ of the null space of $\frac{\partial \tilde{\chi}}{\partial v}(v^*) \in \mathbb{R}^{n_{\tilde{\chi}} \times n_v}$ and let strict complementarity hold. Projecting the Hessian of the Lagrangian function in the null space of the Jacobian of $\tilde{\chi}(v^*)$ ($Z^T\nabla_v^2\mathcal{L}(v^*, \lambda^*, \mu^*)Z$), we obtain the so called reduced Hessian. The following statements hold:*

- *Second Order Necessary Conditions: if v^* is a local minimizer, then the reduced Hessian is positive semidefinite.*
- *Second Order Sufficient Conditions: if the reduced Hessian is positive definite, v^* is a local minimizer unique in its neighborhood.*

With the introduction of Theorem 1 and Theorem 2 we have presented all the elements to be able to distinguish a local minimizer; in Section A.2 we will discuss how to approach these conditions from a numerical point of view.

A.2 NLP Solution via Interior-Point Method

As seen, inequality constraints lead to non-smooth complementary conditions; this means that standard Newton's method cannot be applied directly to solve KKT problems. In this section we will show a method to deal with this kind of problems, the basic idea is to substitute non-smooth conditions with differentiable functions that, at each step of an iterative process, are reshaped in order to give a better approximation of the original constraints. The family of methods we present is called interior-point method and, together with *Sequential Quadratic Programming*, it is the most common way to numerically solve NLPs.

We start analyzing the complementarity conditions of the KKT problem, that is proposed in a less compact version here

$$\begin{aligned}
&\nabla\phi(v^*) + \nabla\chi(v^*)\lambda^* + \nabla\psi(v^*)\mu^* = 0,\\
&\chi(v^*) = 0,\\
&\psi(v^*) \leq 0,\\
&\mu^* \geq 0,\\
&\mu_i^*\psi_i(v^*) = 0 \quad for \quad i \in \{1, 2, ..., n_\psi\}.
\end{aligned}$$

We can think to approximate these conditions with a single set of smooth equations: $\mu_i^*\psi_i(v^*) + \tau = 0$ with $\tau > 0$. These are hyperbolas in the $(\psi_i(v^*), \mu_i^*)$ plane that (for $\tau \to 0$) tend to the L-shaped sets of the the complementarity conditions.

In order to see how the hyperbola approximation has changed the form of our NLP we have to make some operations: expand the last term of the first KKT condition

$$\nabla\psi(v^*)\mu^* = \sum_{i=1}^{n_\psi} \mu_i^* \nabla\psi(v^*)[:, i] = \sum_{i=1}^{n_\psi} \mu_i^* \nabla\psi_i(v^*),$$

substitute the hyperbola approximation of μ_i^* in the latter equation

$$\sum_{i=1}^{n_\psi} \mu_i^* \nabla\psi_i(v^*) = -\tau \sum_{i=1}^{n_\psi} \frac{1}{\psi_i(v^*)} \nabla\psi_i(v^*),$$

noting that $\frac{1}{\psi_i(v^*)}\nabla\psi_i(v^*) = \nabla \log(-\psi_i(v^*))$, we have

$$\nabla\psi(v^*)\mu^* = -\tau \sum_{i=1}^{n_\psi} \nabla \log(-\psi_i(v^*)),$$

gathering this expression with $\nabla\phi(v^*)$ we have

$$\nabla\phi(v^*) + \nabla\psi(v^*)\mu^* = \nabla(\phi(v^*) - \tau \sum_{i=1}^{n_\psi} \log(-\psi_i(v^*))).$$

Finally we can state that the KKT conditions for the above NLP are well approximated (for small values of τ) by the first order optimality conditions of the following problem.

Problem 2 (Barrier Problem)**.**

$$\begin{aligned} &\underset{v}{\textit{minimize}} \quad \hat{\phi}(v,\tau) = \phi(v) - \tau \sum_{i=1}^{n_\psi} \log(-\psi_i(v)) \\ &\textit{subject to} \quad \chi(v) = 0 \end{aligned}$$

The approach to this new problem is the following: we solve it for a big value of τ (in order to have a slightly nonlinear problem), with this solution we initialize a new search using a smaller value of τ, and we iterate until τ is small enough and v^* is reached. A key point of the solution of this problem is to ensure that iterations don't jump to the second branches of the hyperbolas (this is generally achieved by shortening steps).

To conclude this section, we have to explain how to solve each step (one for every value of τ) of Problem 2: every step can be classified as a nonlinear optimization with equality constraints only. We start writing the first order conditions of optimality for these problems

$$\begin{aligned} &\nabla_v \hat{\mathcal{L}}(v,\tau,\lambda) = \nabla_v \hat{\phi}(v,\tau) + \nabla\chi(v)\lambda = 0, \\ &\chi(v) = 0. \end{aligned}$$

Starting from an initial guess $[v_0^T \quad \lambda_0^T]^T$, Newton's method generates a sequence of iterations (denoted by the index k) by linearizing the previous equations.

$$\begin{bmatrix} \hat{B}(v_k,\tau,\lambda_k) & \nabla\chi(v_k) \\ \nabla\chi(v_k)^T & 0 \end{bmatrix} \begin{bmatrix} v - v_k \\ \lambda - \lambda_k \end{bmatrix} + \begin{bmatrix} \nabla_v \hat{\mathcal{L}}(v_k,\tau,\lambda_k) \\ \chi(v_k) \end{bmatrix} =$$
$$\begin{bmatrix} \hat{B}(v_k,\tau,\lambda_k) & \nabla\chi(v_k) \\ \nabla\chi(v_k)^T & 0 \end{bmatrix} \begin{bmatrix} v - v_k \\ \lambda \end{bmatrix} + \begin{bmatrix} \nabla_v \hat{\phi}(v_k,\tau) \\ \chi(v_k) \end{bmatrix} = 0$$

where the matrix on the left is called *KKT matrix* and $\hat{B}(v_k,\tau,\lambda_k) = \nabla_v^2 \hat{\mathcal{L}}(v,\tau,\lambda)$ is introduced to underline the fact that in many circumstances using an approximation of the Hessian of the Lagrangian might be very convenient.

The last equation is linear in the variable $[v^T \quad \lambda^T]^T$ and it can be easily solved inverting the KKT matrix.

A.3 Algorithmic Differentiation

As we have seen in the previous sections of this appendix, differentiation is a fundamental process in numerical optimization. Common ways to compute derivatives (such as *Computer Algebra Systems* or *Finite Differences*) can become too slow or too inaccurate as the expressions get longer. In this section we present the basic idea behind Algorithmic Differentiation methods used by numeric optimization software such as CasADi.

Let us consider a function $\rho(v) : \mathbb{R}^{n_v} \to \mathbb{R}^{n_\rho}$, that is composed by a sequence of n_σ elementary operations $\sigma_i(v)$ for $i \in \{0, 1, ..., n_\sigma - 1\}$. We denote the inputs as $v_1, v_2, ..., v_{n_v}$ and the result of the i^{th} intermediate elementary operation as v_{n_v+i+1} $(= \sigma_i(v_1, v_2, ..., v_{n_v+i}))$.

In order to compact results, we introduce augmented vectors that collects the inputs and the first intermediate results:

$$\tilde{v}_0 = v = \begin{bmatrix} v_1 \\ \vdots \\ v_{n_v} \end{bmatrix}, \ \tilde{v}_1 = \begin{bmatrix} v_1 \\ \vdots \\ v_{n_v} \\ v_{n_v+1} \end{bmatrix}, \ ..., \ \tilde{v}_{n_\sigma} = \begin{bmatrix} v_1 \\ \vdots \\ v_{n_v} \\ \vdots \\ v_{n_v+n_\sigma} \end{bmatrix}.$$

In the same way we define the augmented elementary function:

$$\tilde{\sigma}_i(\tilde{v}_i) = \begin{bmatrix} v_1 \\ \vdots \\ v_{n_v+i} \\ \sigma_i(v_1, v_2, ..., v_{n_v+i}) \end{bmatrix}.$$

Using this representation, we can exploit the chain rule of differentiation to achieve an useful result. Defining C as a selection matrix, that extract form the last result $(\tilde{v}_{n_\sigma})$ the desired outputs, we can put our initial expression in the following form:

$$\rho(v) = C \cdot \tilde{\sigma}_{n_\sigma-1}(\tilde{\sigma}_{n_\sigma-2}(...(\tilde{\sigma}_1(\tilde{\sigma}_0(v))))).$$

At this point we are interested in the computation of the Jacobian matrix of $\rho(v)$. Since the obtained expression collects every elementary operation one inside the other, the derivative of the the function is obtained by the multiplication of the derivatives of each elementary function:

$$\frac{\partial \rho}{\partial v} = C \cdot \frac{\partial \tilde{\sigma}_{n_\sigma-1}}{\partial \tilde{v}_{n_\sigma-1}} \cdot \frac{\partial \tilde{\sigma}_{n_\sigma-2}}{\partial \tilde{v}_{n_\sigma-2}} \cdot ... \cdot \frac{\partial \tilde{\sigma}_1}{\partial \tilde{v}_1} \cdot \frac{\partial \tilde{\sigma}_0}{\partial \tilde{v}_0}.$$

This result already simplify and make systematic the operation of differentiation but, if we examine how each Jacobian matrix is composed, we can see that the generic $\frac{\partial \tilde{\sigma}_i}{\partial \tilde{v}_i}$ has the following structure:

$$\frac{\partial \tilde{\sigma}_i}{\partial \tilde{v}_i} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ * & * & \cdots & * \end{bmatrix}$$

where the row composed by $*$ has at maximum two non-zero elements (since each elementary operation has maximum two inputs).

The last result explains why AD is so efficient: every step of multiplication consists in a sequence of computational cost free operations (where 0 or 1 elements are involved) and a multiplication of a vector (composed by two non-zero elements only) for a very sparse matrix.

The product of the Jacobian matrices can be evaluated in the *Forward Mode* (FM) or *Backward Mode* (BM), depending on the order with which the multiplications are performed.

Introducing a so called *seed vector* $p \in \mathbb{R}^{n_v}$ (whose role is to select the "direction" of the derivative), the FM starts from the multiplication $\frac{\partial \tilde{\sigma}_0}{\partial \tilde{v}_0} p$ and makes a recursion whose result is the directional derivative of $\rho(v)$: $\frac{\partial \rho}{\partial v} p$. It's important to note that in this case if we want to obtain the full Jacobian matrix $\frac{\partial \rho}{\partial v}$ we have to call the algorithm n_v times, using at every sweep a seed vector corresponding to a different unit vector in $\mathbb{R}^{n_v}$ (it can be shown that the computational cost of the full Jacobian is directly proportional to n_v).

On the other hand, BM starts the recursion from $\lambda^T C$ (where $\lambda \in \mathbb{R}^{n_\rho}$ is the seed vector that every sweep combine the rows of the full Jacobian) to reach the result $\lambda^T \frac{\partial \rho}{\partial v}$. Note that this time if we want to obtain $\frac{\partial \rho}{\partial v}$ we have to call the algorithm n_ρ times, using at every sweep a seed vector corresponding to a different unit vector in $\mathbb{R}^{n_\rho}$ (this time the computational cost is directly proportional to n_ρ).

The last observations explain why BM is so efficient in numerical optimization problems: thinking for example at the Lagrangian function (that is a scalar function: $n_\rho = 1$) of a system with a huge number of states ($n_v \gg 1$), BM involve a great computational economy. As a drawback BM has the necessity to store all the intermediate variables and partial derivatives, and it is also true that not all the times we have a function where $n_v > n_\rho$.

Bibliography

[1] J. T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming.* SIAM, 2001.

[2] M. Diehl. (2014) Lecture notes on optimal control and estimation. [Online]. Available: `http://syscop.de/wp-content/uploads/2015/03/oce_script.pdf`.

[3] H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," in *9th IFAC World Congress*, 1984.

[4] T. Tsang, D. Himmelblau, and T. Edgar, "Optimal control via collocation and nonlinear programming," *Int. Journal of Control (IJC)*, vol. 21, pp. 763–768, 1975.

[5] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *Int. Journal of Robotics Research (IJRR)*, vol. 33, no. 1, pp. 69–81, 2014.

[6] W. Xi and C. Remy, "Optimal gaits and motions for legged robots," in *IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014, pp. 3259–3265.

[7] M. Gabiccini, A. Artoni, G. Pannocchia, and J. Gillis, "A computational framework for environment-aware robotic manipulation planning," in *International Symposium on Robotics Research (ISRR)*, 2015.

[8] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, 2014.

[9] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," in *ACM Transactions on Graphics*, 2012.

[10] I. Mordatch, Z. Popović, and E. Todorov, "Contact-invariant optimization for hand manipulation," in *Eurographics/ACM Symposium on Computer Animation*, 2012.

[11] I. Mordatch and E. Todorov, “Combining the benefits of function approximation and trajectory optimization,” in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[12] D. Pardo, L. Möller, M. Neunert, A. Winkler, and J. Buchli, “Evaluating direct transcription and nonlinear optimization methods for robot motion planning,” 2015, arXiv:1504.05803v1 [math.OC].

[13] S. Karaman and E. Frazzoli, “Sampling-based Algorithms for Optimal Motion Planning,” in *International journal of robotics research*, vol. 30, no. 7, pp. 846–894, June 2011.

[14] S. M. LaValle, *Planning Algorithms.* Cambridge University Press, 2006.

[15] P. de Leva, “Adjustments to Zatsiorsky-Seluyanov’s segment inertia parameters,” *Journal of Biomechanics*, vol. 29, no. 9, pp. 1223 – 1230, 1996.

[16] J. Andersson, “A General-Purpose Software Framework for Dynamic Optimization,” PhD thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium, October 2013.

[17] A. Wächter and L. T. Biegler, “On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, pp. 25–57, 2006.

[18] I. Wolfram Research, *Mathematica*, ser. Version 10.2. Champaign, Illinois: Wolfram Research, Inc., 2015.

[19] J. Nocedal and S. J. Wright, *Numerical Optimization.* New York: Springer Verlag, 2006.