Report about Dynamic Complementarity Conditions and Whole-Body Trajectory Optimization for Humanoid Robot Locomotion

Saverio Taliani, Pietro Bonsanto

February 2023

Contents

Abstract

In this report we focus on the main topics covered in the presentation, paying attention to the mathematical structure of the problem. Here we assume no knowledge about the models as we will disclose the contents of the paper of interest. We will address the problem of locomotion control for a humanoid robot, i.e. a nonlinear optimization problem, both for the *real* problem and for a *toy* problem (i.e. a mass in free falling to simulate the robot stepping). We will address the model both for the robot and for the constraints, the main issues related with them and some generalities about the control techniques hereby used, and a particular issue when controlling the whole dynamics. The discussion will be validated through a series of simulations.

1 Generalities About the Problem

As anticipated before, the main problem is related to the possibility of planning the motion for a humanoid robot. This kind of locomotion is achieved by means of an optimization problem of high dimension. In order to get acquainted with the problem, we will first need to model the robot itself and its interaction with the environment. The latter will be asserted via a particular kind of constraints :

- Complementarity conditions
- Dynamic complementarity conditions
- Hyperbolic complementarity conditions

As far as the motion generation, i.e. the so called *trajectory optimization* , it is in charge of generating foothold trajectories using the well-known receiding horizon principle. When dealing with the problem, it is used a hiearchical control architecture, in which each loop receives inputs from the robot and the environment and provides references to the loop next; the inner the layer, the shorter the time horizon used to evaluate the outputs. This hierarchical control architecture is composed of 3 loops:

- trajectory optimization
- simplified model control
- whole-body quadratic programming control loop

1.1 Trajectory Optimization Loop

This loop deals with the generation of foothold trajectories, and has as output the simplified model control loop. Several solutions exist in order to deal with this problem, but all of them have in common the fact that planning is focused on the Center Of Mass state. Since this, the main issue in related to the nonlinearitiy of the angular momentum dynamics.

1.2 Simplified Model Control Layer

This layer aims to find feasible trajectories for the robot CoM along the walking path. In general this is a quite difficult problem, but with the simplifying hypoteses that the Com resides on a plane at a fixed height and assuming constant angular momentum, we can derive simpler control laws based on the linear inverted pendulum. As a remark, it is important to state that there are several emerging strategies relying on the combination of the simplified model control into the trajectory optimization layer in order to achieve multi-objectives with a single optimization problem. As far as the quadratic programming loop is concerned, instead of considering the entire robot model in a single optimal control problem, it is possible to plan trajectories for the centroidal and joint quantities separately, iterating until reaching a consensus between the two solution trajectories.

1.3 Approach to the Problem

In order to deal with the proposed problem, the solution relies on the full robot kinematics and centroidal momentum, together with an alternative formulation for the classical complementarity conditions (introducing DCCs). This alternative, namely the Dynamic Complementarity Conditions, can be embedded into a whole-body nonlinear optimization framework aimed at finding feasible footsteps and whole-body trajectories. The latter is computed using the receiding horizon principle.

2 Humanoid Robot Modelling

2.1 Dynamic Model

For the time being, we consider the following model:

$$M(q)\dot{\nu} + C(q,\nu)\dot{q} + G(q) = \begin{pmatrix} 0_{6xn} \\ I_n \end{pmatrix} \tau_s + \sum_{k=1}^{n_c} J_{c_k\,k}^T f \tag{1}$$

Where $M \in \mathcal{R}^{(n+6)\times(n+6)}$ is the mass matrix, $C \in \mathcal{R}^{(n+6)\times(n+6)}$ refers to Coriolis and centrifugal effects, $G \in \mathcal{R}^{n+6}$ is the gravity term, $\tau_s \in \mathcal{R}^n$ is the torque, and $f \in \mathcal{R}^6$ refers to the external wrench applied by the envinroment to the robot:

$$f = \begin{pmatrix} f \\ \tau \end{pmatrix} \tag{2}$$

with $f, \tau \in \mathcal{R}^3$ contact forces and torques. In order to decouple joint and base accelerations, it is possible to apply a coordinate transformation in the state space. In these new coordinates, the first six rows of (1) correnspond to the centroidal dynamics.

2.2 Centroidal Dynamics

Centroidal Dynamics is of a preminent importance when dealing with humanoid robots, because it enables us to treat the whole body as a system and only consider the external interaction. First, define the CoM $x \in \mathcal{R}^3$ as the weighted average of all the link's CoM position:

$$x = H_B \sum_i H_i^i p_{CoM} \frac{m_i}{m} \tag{3}$$

where p_{CoM} is the CoM position of the ith link expressed w.r.t. the ith coordinate system. In order to introduce the *centroidal dynamics* we first define the frame $\bar{G}$ with origin in the CoM, and define with ${}_G h \in \mathcal{R}^6$ the robot total momentum expressed w.r.t. $\bar{G}$:

$${}_{\bar{G}}h = \begin{pmatrix} {}_{\bar{G}}h^p \\ {}_{\bar{G}}h^w \end{pmatrix} \tag{4}$$

with ${}_{\bar{G}}h^p \in \mathcal{R}^3$ and ${}_{\bar{G}}h^w \in \mathcal{R}^3$ linear and angular momentum. Moreover, the CoM is such that:

$$\dot{x}_{CoM} = \frac{1}{m}{}_{\bar{G}}h^p \tag{5}$$

One obtains:

$${}_{\bar{G}}h = J_{CMM}\nu \tag{6}$$

where $J \in \mathcal{R}^{(6\times n)}$ is the centroidal momenutm matrix. Its dynamics can be obtained by differentiating (6) w.r.t. time , yielding

$$\bar{G}\dot{h} = J_{CMM}\dot{\nu} + \dot{J}_{CMM}\nu \tag{7}$$

2.3 Complementarity Conditions

In order to model the footstepping, we need a tool to model the contact surface between the foot and the ground. In order to achieve this, we introduce the Complementarity Conditions. First, we assume the foot is composed by a set of points on which the environment will apply a certain force. On one hand we have that the contact points are not supposed to penetrate the walking ground, and on the other hand we know that the resulting force from the interaction, namely a reaction (thus, normal w.r.t. the walking ground) force, is non-negative. Its value is different from zero only if the associated contact point is in contact with the walking surface, i.e.

$$h({}_ip)n({}_ip)_i^T f = 0 \tag{8}$$

which is exactly the conditon we were looking for, in which h is a function of the contact points, $n({}_ip)_i^T f$ is the component of the force normal to the ground. However, this formulation yields some numerical issues, so it can be relaxed bounding it with a infinitesimal quantity ϵ:

$$h({}_ip)n({}_ip)_i^T f \leq \epsilon \tag{9}$$

which we refer to as *relaxed complementarity*

2.4 Dynamic Complementarity Conditions

The previous conditions can be enforced using a Baumgarte stabilization method (see Appendix). In order to achieve convergence to 0, we set the so-called dynamical constraints:

$$\frac{d}{dt}(h({}_ip)n({}_ip)_i^T f) = -K_{bs}(h({}_ip)n({}_ip)_i^T f) \tag{10}$$

with $K_{bs} \in \mathcal{R}^n$ positive gain. When evaluating the derivative ad the left-hand side of the previous equations, one has:

$$\frac{d}{dt}(\circ) = \frac{d}{dt}(h({}_ip)n({}_ip)_i^T f) +$$

$$+h({}_ip)_i f^T \frac{d}{dt}(n({}_ip)) + h({}_ip)n({}_ip)_i^T \dot{f} \tag{11}$$

We can substitute the time derivative of $h(\circ)$ and $n(\circ)$ with the kinematic derivatives and use the implicit function theorem, together with the substitution

$$\zeta := \frac{\partial}{\partial_i p}(h({}_ip))_i \dot{p} n({}_ip)_i^T f$$

$$+h({}_ip)_i f^T \frac{\partial}{\partial_i p}(n({}_ip))_i \dot{p} + h({}_ip)n({}_ip)_i^T \dot{f} \tag{12}$$

yielding the condition:

$$\zeta \leq -K_{bs}(h({}_ip)n({}_{i_p})_i^T f) \tag{13}$$

In an analogous way as in (9) we finally add a relaxation via positive number $\epsilon \in \mathcal{R}^+$ to increase the feasibility region:

$$\zeta \leq -K_{bs}(h({}_ip)n({}_{i_p})_i^T f) + \epsilon \tag{14}$$

which is the final version of the dynamic complementarity conditions

2.5 Hyperbolic Complementarity Conditions

In order to impose (9) we can eforce the following set of constraints on the force derivative:

$$-M_f \leq \dot{f} \leq M_f, h(p) = 0 \tag{15}$$

$$\dot{f} = -K_f f, h(p_i) \neq 0 \tag{16}$$

which means that when the point is in contact ${}_i\dot{f}$ can assume all the values in the interval $[-M_f, M_f]$, with $M_f \in \mathcal{R}^3$. The previous condition can be rewritten in terms of the following binary function $\delta^*({}_ip)$:

$$\delta^*({}_ip) = \begin{cases} 1 & h({}_ip) = 0 \\ 0 & h({}_ip) \neq 0 \end{cases} \tag{17}$$

as the following set of inequalities:

$$-K_f(1 - \delta^*(p_i))f_i - \delta^*(p_i)M_f \leq \dot{f} \tag{18}$$

$$-K_f(1 - \delta^*(p_i))f + \delta^*(p_i) \geq \dot{f} \tag{19}$$

We introduce the hyperbolic secant as the continuous approximation of the function $\delta^*({}_ip)$, namely:

$$\delta({}_ip) = sech(k_h h({}_ip)) \tag{20}$$

with k_h scaling factor. If we assume that the force decreases at an higher rate than the one in (18)-(19), we can rewrite :

$$-M_f \leq_i \dot{f} \leq -K_f(1 - \delta({}_ip))_i f + \delta({}_ip)M_f \tag{21}$$

Finally, if we apply one of these equations only to the force component normal to the ground, we derive the condition:

$$-e_3^T M_f \leq e_3^T {}_i\dot{f} - K_{f,z}(1 - \delta({}_ip))n({}_ip)_i^T f + \delta(i_p)e_3^T M_f \geq e_3^T {}_i\dot{f} \tag{22}$$

3 DCCs Based Nonlinear Trajectory Planning

In this section we go into details of the optimization problem. The problem of trajectory planning is modeled as an optimization problem in which the objective is composed of a variety of tasks posed to make the robot reach a desired position without specifying the sequence of steps. In fact the solution results to be a feasible trajectory for the whole robot. First of all, we assume to have full control over the derivatives of the contact point's positions and forces:

$${}_i\dot{p} = u_{ip} \tag{23}$$

$${}_i\dot{f} = u_{if} \tag{24}$$

where we denote with $u_{ip}, u_{if} \in \mathcal{R}^3$ the control inputs for the contact points. As far as the latter are concerned, we want them to remain in the same surface with a certain fixed relative distance and to be within the reachable workspace of the robot legs. Both the conditions can be achieved imposing the following algebraic condition:

$${}_ip =^I H_{foot}^{foot}{}_ip \tag{25}$$

where ${}_i^{foot}p$ is the position of the contact point within the foot surface. As far as the robot control is concerned, we assume that the joint values can be assigned at will, which can be rendered via the following:

$$\dot{s} = u_s \tag{26}$$

This assumption requires and additional control loop: joint velocities with contact vertex positions and forces, are considered as reference to an inner whole-body controller. The las missing piece is about the rotation included in q, which is vectorized using the quaternion parametrization, called ${}^I\rho_B \in \mathcal{H}$. The dynamics of the base can be exploited via the following dynamic equations:

$${}^I\dot{p}_B = {}^I R_B^B v_{I,B} \tag{27}$$

$${}^I\dot{\rho}_B = u_\rho \tag{28}$$

with ${}^B v_{I,B} \in \mathcal{R}^3, u_\rho \in \mathcal{R}^4$ and $u_s \in \mathcal{R}^n$ are control inputs.

3.1 Planar DCC

In order to prevent the contact points to move on the walking plane when in contact with the ground, we can either use the relaxed planar complementarity conditions (see Appendix) or proceed by limiting the effect of the control input u_p along the planar directions, and forcing these projections to zero when $h(i_p) = 0$. This is achieved using the *hyperbolic tangent*:

$$t(i_p)_i^T \dot{p} = tanh(k_t h({}_i p))[e_1 e_2]^T u_{i,p} \tag{29}$$

In this framework, the latter can be used to rewrite (23) as follows:

$${}_i\dot{p} = \tau(i_p) u_{ip} \tag{30}$$

where τ is defined as

$$\tau({}_i p) = {}^I R_{plane} diag \begin{pmatrix} tanh(k_t h({}_i p)) \\ tanh(k_t h({}_i p)) \\ 1 \end{pmatrix} \tag{31}$$

This last extension provides many advantages, related both to the 0 bound on the velocity when approaching the contact surface and to the possibility of controlling the height of the contact points when swinging

3.2 Momentum Control

In this section we describe the effect of the forces, exerted by the contact points on the environment, on the robot motion using the centroidal dynamics:

$$\bar{G}\dot{h} = m\bar{g} + \sum_i \begin{pmatrix} I_3 \\ ({}_i p - x) \end{pmatrix}_i f \tag{32}$$

We need to make sure that the CoM position obtained by integrating (5) correnspondsto the one obtained via joint variables. This is ensured by the following equation:

$$x = CoM({}^I\rho_B, {}^I\rho_B, s) \tag{33}$$

which is the function mapping base pose and joints position to the CoM position. Since the latter links the linear momentum and the joint variables, we need to link the angular momentum and the joints evolution. This is achieved by means of the CMM J_{CMM}, yelding:

$$ {}_{\bar{G}}h^w = \begin{pmatrix} 0_{3x3} & I_3 \end{pmatrix} J_{CMM} \nu \in \mathcal{R}^3 \tag{34} $$

Note that ν contains the base angular velocity ${}^B w_{I,B}$, which can be substituted with the quaternion derivative using the mapping $\mathcal{G}$:

$$ \mathcal{G}({}^I\rho_B) = \begin{pmatrix} {}^I\rho_{B,1} & {}^I\rho_{B,0} & {}^I\rho_{B,3} & {}^I\rho_{B,2} \\ {}^I\rho_{B,2} & {}^I\rho_{B,3} & {}^I\rho_{B,0} & {}^I\rho_{B,1} \\ {}^I\rho_{B,3} & {}^I\rho_{B,2} & {}^I\rho_{B,1} & {}^I\rho_{B,0} \end{pmatrix} \tag{35} $$

and such that

$$ {}^B w_{I,B} = 2\mathcal{G}({}^I\rho_B) u_\rho \tag{36} $$

3.3 Complete Differential-Algebraic Model

As a summary, in the sequel all the differential equations and algebraic conditions are summarized, yielding an *inequality constrained differential-algebraic system of equations*

- Dynamical constraints:

$$ {}_i\dot{f} = u_{if} \tag{37} $$

$$ {}_i\dot{p} = \tau({}_ip) u_{ip} \tag{38} $$

$$ {}_{\bar{G}}\dot{h} = m\bar{g} + \sum_i \begin{pmatrix} I_3 \\ ({}_ip - x) \end{pmatrix}_i f \tag{39} $$

$$ \dot{x} = \frac{1}{m}({}_{\bar{G}}h^p) \tag{40} $$

$$ {}^I\dot{p}_B = {}^I R_B^B v_{I,B} \tag{41} $$

$$ {}^I\dot{\rho}_B = u_p \tag{42} $$

$$ \dot{s} = u_s \tag{43} $$

- Equality Constraints

$$ {}_ip = {}^A H_{foot}^{foot} {}_ip \tag{44} $$

$$ x = CoM({}^I\rho_B, {}^I\rho_B, s) \tag{45} $$

$$ {}_{\bar{G}}h^w = \begin{pmatrix} =_{3x3} & I_3 \end{pmatrix} J_{CMM} \begin{pmatrix} {}^B v_{I,B} \\ 2\mathcal{G}({}^I\rho_B) u_\rho \\ u_s \end{pmatrix} \tag{46} $$

$$ ||{}^I\rho_B||^2 = 1 \tag{47} $$

- Inequality constraints, applied for each contact point:

$$ n({}_ip)_i^T f \geq 0 \tag{48} $$

$$ ||t({}_ip)_i^T f|| \leq \mu_s n({}_ip)_i^T f \tag{49} $$

$$ -M_V \leq u_{ip} \leq M_V \tag{50} $$

$$ -M_f \leq u_{if} \leq M_f \tag{51} $$

$$ h({}_ip) \geq 0 \tag{52} $$

3.4 About Constraints during motion

As an additional remark, it is important to highlight that we want to prevent the robot legs to collide with each other when stepping. Since self-collisions constraints are hard to implement, we avoid the problem avoiding cross-steps. This means that we are assuming that the frame attached to the right foot has the positive y-direction pointing towards left. In this case, it is enough to impose the y-component of the ${}^{r}x_l$ (the relative position of the left foot expressed in the right foot frame) to be greater than a certain quantity, i.e.

$$e_2^{T\,r}x_l \geq d_{min} \tag{53}$$

In order to prevent the swinging leg to cause other self-collisions, we set an upper bound on the difference between the height of the two feet:

$$-M_{hf} \leq e_3^T(p_l - p_r) \leq M_{hf} \tag{54}$$

where we consider the mean position of every contact point for simplicity. Finally, we can add the following additional constraints:

$$x_{zmin} \leq h(x) \tag{55}$$

$$-M_{h_w} \leq_{\bar{G}} h^w \leq M_{h_w} \tag{56}$$

Where the first avoids emergence of solutions that set the CoM position too close or even below the ground, whereas the second sets a bound $M_{h_w} \in \mathcal{R}^3$ on the angular momentum.

3.5 Tasks in Cartesian Space

In order to make the robot approach a desired position, we need to specify the task in *Cartesian Space* In the following we present some different types of task:

- *Contact Point Centroid Position Task* We define as a task the L2 norm of the error between a point attached to the robot and its desired position in an absolute frame. If we choose the CoM as the desired point in space, the robot could lean forward and fall. In order to prevent this to happen, we can locate the target point on the feet instead of the CoM. In particular, we select the centroid of the feet contact point as target:

$$\Gamma_p = \frac{1}{2}||p - p^*||^2_W \tag{57}$$

  where $p = \frac{1}{2}(p_l + p_r)$ and $p^* \in \mathcal{R}^{\in}$ is its desired value.

- *CoM linear velocity task* While walking, we want the robot to keep a constant forward motion. Requiring a constant forward velocity could in fact help preventing to plan two consecutive steps with the same foot. This is achieved if we set

$$\Gamma_{\bar{G}h^p} = \frac{1}{2}||{}_{\bar{G}}h^p - m\dot{x}^*||^2_{W_v} \tag{58}$$

  where the weights W_v allow to select and weight the different directions separately.

- *Foot Yaw Task* This task specifies the angle at which the foot should be oriented w.r.t. the z-axis, i.e. the foot yaw angle. Denote as $\gamma^*_\circ$ the desired yaw angle. We construct the following vector:

$$l^*_\circ = \begin{pmatrix} cos(\gamma^*_\circ) \\ sin(\gamma^*_\circ) \end{pmatrix} \tag{59}$$

Where the vector $l_\circ \in \mathcal{R}^2$ is fixed to the foot and parallel to the foot x-axis and expressed in the I frame. The goal for this task is to align $l_\circ$ to $l^*_\circ$, which is achieved by locally minimizing their cross-product, yielding:

$$\Gamma'_{yaw} = \sum_{l,r} \frac{1}{2} ||(-sin(\gamma^*_\circ)cos(\gamma^*_\circ))l_\circ||^2 \tag{60}$$

The latter can be minimized by shrinking the projection on the xy-plane of the vector attached to the robot foot, but this is undesired because we would set the foot to be perpendicular to the ground. Hence, we consider also a second vector attached to the foot and perpendicular to $l_\circ$ called $l^\perp_\circ$. The final task has the form:

$$\begin{aligned} \Gamma_{yaw} = &\sum_{l,r} \frac{1}{2} || \begin{pmatrix} -sin(\gamma^*_\circ) & cos(\gamma^*_\circ) \end{pmatrix} l_\circ||^2 \\ &+ \sum_{l,r} \frac{1}{2} || \begin{pmatrix} cos(\gamma^*_\circ & sin(\gamma^*_\circ) \end{pmatrix} l^\perp_\circ||^2 \end{aligned} \tag{61}$$

3.6 Regularization Task

Despite having modelled a consistent amount of the system, we still have neither neglected several parts of it nor constrained. By this, we need to introduce regularization tasks that help generating trajectories.

- *Frame Orientation Task* While in motion, we want a generic robot frame to take a desired orientation ${}^I R^*_{frame}$. We weight the distance of the rotation matrix ${}^I \tilde{R}_{frame} = {}^I R^*_{frame}{}^T {}^I R_{frame}$ from the identity. To do this, we convert ${}^I \tilde{R}_{frame}$ into a quaternion and weight the difference from the identity quaternion I_q, thus denoting the Euclidean distance of quaternion as a metric for rotation error:

$$\Gamma_{frame} = \frac{1}{2} ||quat({}^A \tilde{R}_{frame} - I_q||^2 \tag{62}$$

- *Base quaternion Derivative Regularization Task* This task helps us when tracking a desired angular velocity for the base and in regularizing the robot motion. It is defined as follows:

$$\Gamma_{regu_\rho} = \frac{1}{2} ||u_\rho - u^*_\rho||^2 \tag{63}$$

- *Force Regularization Task* While considering foot contact forces independent from each other, they still belong to the same rigid body. So we impose the contact forces in a foot to be as similar as possible, avoiding the use of partial contacts if not strictly needed. The latter is achieved by imposing:

$$\Gamma_{regf} = \sum_{l,r} \sum_{i}^{n_p} \frac{1}{2} ||_i f - diag(_i \alpha^*) \sum_{j}^{n_p} {}_j f||^2_{W_f} \tag{64}$$

where α determines the desired ratio for force i w.r.t. total forces.

- *Joint Regularization Task* Used when aiming to avoid solutions with huge joint variables, imposing

$$\Gamma_{regs} = \frac{1}{2}||\dot{s} + K_s(s - s^*)||^2_{W_j} \tag{65}$$

It is straightforward to notice that the minimum for this cost is achieved when $\dot{s} = -K_s(s - s^*)$ with $K_s \in \mathcal{R}^{nxn}$. When the latter holds, joint values converge exponentially to their desired values s^*

- *Swing Height Task* We want to specify a desired swing height, which can ensure us some level of robustness w.r.t. ground asperity. To this end we define the following task:

$$\Gamma_{swing} = \sum_{l,r}\sum_{i}^{n_p} \frac{1}{2}(e_3^T{}_i p -_s h^*)^2 || \begin{pmatrix} e_1 & e_2 \end{pmatrix}^T u_{i,p}||^2 \tag{66}$$

which penalizes the distance between the z-component of each contact point position from a desired height ${}_s h^*$

- *Contact Control Regularization Tasks* Even if we have assumed the contact point velocities and force derivatives to be control inputs, we want to avoid the planner to use them always at the limit. So, we add the following tasks:

$$\Gamma_{regu_p} = \sum_{l,r}\sum_{i}^{n_p} \frac{1}{2}||u_{ip} - u^*_{ip}||^2_{W_{up}} \tag{67}$$

$$\Gamma_{regu f} = \sum_{l,r}\sum_{i}^{n_p} \frac{1}{2}||u_{if} - u^*_{if}||^2_{W_{uf}} \tag{68}$$

3.7 Complete Optimal Control Problem

Taking into account the equations and the task so far presented, we can formulate an optimal control problem which complete formulation is the following:

$$\min_{\chi,\mathcal{U}} w^T \begin{pmatrix} \Gamma_p \\ \Gamma_{\bar{G}h^p} \\ \Gamma_{frame} \\ \Gamma_{regu_\rho} \\ \Gamma_{reg,f} \\ \Gamma_{regs} \\ \Gamma_{swing} \\ \Gamma_{yaw} \\ \Gamma_{regu_\rho} \\ \Gamma_{regu_f} \end{pmatrix} \tag{69}$$

subject to the following constraints:

$$\dot{\chi} = f(\chi,\mathcal{U}) \tag{70}$$

$$l \leq g(\chi,\mathcal{U}) \leq u \tag{71}$$

$$e_2^{T_r} x_l \geq d_{min} \tag{72}$$

$$x_{zmin} \leq h(x) \tag{73}$$

$$-M_{h_w} \leq_{\bar{g}} h^w \leq M_{h_w} \tag{74}$$

$$-M_{hf} \leq e_3^T(p_l - p_r) \leq M_{hf} \tag{75}$$

Where we have:

$$\chi = \begin{pmatrix} {}_{i}f \\ {}_{i}p \\ \vdots \\ {}_{\bar{G}}h \\ x \\ {}^{I}p_B \\ {}^{I}\rho_B \\ s \end{pmatrix}, \mathcal{U} = \begin{pmatrix} u_{if} \\ u_{ip} \\ \vdots \\ {}^{B}v_{I,B} \\ u_\rho \\ u_s \end{pmatrix} \tag{76}$$

3.8 Issues in Implementation

The optimal control problem hereby presented is designed in such a way that none of all constraints is task-specific. As a direct consequence, it is very important to define carefully the cost function, recalling the solution is a trade-off between the tasks. On the other hand, the perfect knowledge of the model enables us to achieve walking motions without specifying a desired CoM trajectory. We use the receding horizon principle in order to plan for trajectories longer than the prediction horizon. The receding horizon principle is a decision-making framework used when dealing with particular kind of optimization problems which involves making decisions based on a finite planning horizon that is continually adjusted as new information about the dynamics becomes available.

According to the receding horizon principle, the planner establishes a plan for a specific period, typically called the "horizon," based on the current information available. As time progresses, the planner receives new information about the system or environment they are operating in, and the horizon is shifted forward to account for this new information.

This continual adjustment of the planning horizon allows for more accurate and adaptable decision-making, as decisions can be made based on the most up-to-date information available. The main issue when dealing with this principle is related to the so-called *procastination* phenomena. Procrastination refers to the tendency to delay making decisions until the last possible moment. This can happen because the planner is waiting for more information, is unsure about the best course of action, or is simply avoiding making a commitment. Procrastination can be problematic in the context of the receding horizon principle because it can lead to suboptimal solutions. The principle involves continually adjusting the planning horizon based on new information, and delaying decisions can limit the available time for adjustment. In addition, decisions made in haste are more likely to be based on incomplete or inaccurate information, which can further decrease the effectiveness of the decision-making process. In this case, procastination reflects on the fact that the solver delays in actuating motion, resulting in no motion at all. Moreover, due to the high number of constrains related to the problem ,the latter is nonconvex and the solver may be able to find only suboptimal solutions.

4 Validation on a Toy Problem

In order to evaluate the well-posedness of our problem, we use the model of a point mass in free falling vertically approaching the ground, to model the footstepping of the humanoid robot. The simplified model has the following dynamics:

$$\dot{x}_m = v_m \tag{77}$$

$$\dot{v}_m = g + \frac{1}{m}(f_m + p_m) \tag{78}$$

$$\dot{f}_m = u_f \tag{79}$$

where $v_m, g, f_m, p_m, u_f \in \mathcal{R}$ are respectively the mass velocity, the gravity acceleration, the contact force, the propeller force and the contact force derivative. In this case, the objective to be minimized is the propeller. The optimal control formulation for this problem is:

$$\min p_m \tag{80}$$

subject to:

$$\dot{x_m} = v_m \tag{81}$$

$$\dot{v_m} = g + \frac{1}{m}(f_m + p_m) \tag{82}$$

$$\dot{f_m} = u_f \tag{83}$$

$$x_m \geq 0 \tag{84}$$

$$f_m \geq 0 \tag{85}$$

$$-M_{uf} \leq u_f \leq M_{uf} \tag{86}$$

4.1 Solution to the Toy Problem

The simulation for the toy problem is implemented in MATLAB using *casadi* library as seen in the paper. The specification followed are:

$$x_0 = 0.1[m] \tag{87}$$

$$v_0 = 0[\frac{m}{s}] \tag{88}$$

$$m = 0.2[kg] \tag{89}$$

For the simulations, make reference to the plots in the presentation.

5 Appendix

5.1 Baumgarte Stabilization method

The Baumgarte stabilization method is a technique used in numerical simulation to stabilize the behavior of a system by introducing a constraint penalty term into the equations of motion. however, constraints can cause numerical instability in the simulation, leading to unphysical behavior. The Baumgarte stabilization method addresses this issue by adding a penalty term to the equations of motion that damps the constraint violation.

The penalty term is a function of the constraint violation and its time derivative, and it acts as a corrective force to stabilize the system. The strength of the penalty term is controlled by a user-defined parameter, called the Baumgarte parameter, which determines the balance between the stability of the simulation and the accuracy of the constraint.

The Baumgarte stabilization method is widely used in various fields, including mechanical and aerospace engineering, robotics, and computational physics. It provides an effective and computationally efficient way to simulate systems with constraints accurately while maintaining numerical stability. In practice, for a Multi-Body System (MBS), we have that if we are able to model a generic constraint via a set of m algebraic constraints of the type:

$$\Phi(q,t) = 0 \tag{90}$$

in which q is the vector of generalized coordinates. The general equation of motion for the constrained robot can be rewritten as:

$$M\ddot{q} + \Phi_q^T \lambda = n \tag{91}$$

where n is the generalized force vector that contains the gravitational, Coriolis and Centrifugal forces. In order to progress with the solution, we need the constraint velocity and acceleration equations. To this end, differentiating the latter w.r.t. time yields:

$$\Phi_q \dot{q} = -\Phi_t = v \tag{92}$$

In which v represents a term that contains the partial derivatives of Φ w.r.t. time. If we differentiate again, what we get is:

$$\Phi_q \ddot{q} = -(\Phi_q \dot{q})_q \dot{q} - 2\Phi_{qt}\dot{q} - \Phi_{tt} = \gamma \tag{93}$$

(87),(89),(90) must be satisfied during the motion. To this purpose, we can first substitute (90) in the motion equation and rewrite in matrix form as:

$$\begin{pmatrix} M & \Phi_q^T \\ \Phi_q & 0 \end{pmatrix} \begin{pmatrix} \ddot{q} \\ \lambda \end{pmatrix} = \begin{pmatrix} g \\ \gamma \end{pmatrix} \tag{94}$$

which is the resulting system for $\ddot{q}$ and λ. The Baumgarte Stabilization methos (BSM) helps in keeping the constraints violations under control, allowing violation before the corrective action takes place. The goal of this method is to replace eq (90) with the following:

$$\ddot{\Phi} + 2\alpha\dot{\Phi} + \beta^2\Phi = 0 \tag{95}$$

where the terms $2\alpha\dot{\Phi}$ and $\beta^2\Phi$ in the latter represent corrective terms. Due to this approach, the equations for a system subjected to constraints is of the form:

$$\begin{pmatrix} M & \Phi_q^T \\ \Phi_q & 0 \end{pmatrix} \begin{pmatrix} \ddot{q} \\ \lambda \end{pmatrix} = \begin{pmatrix} g \\ \gamma - 2\alpha\dot{\Phi} - \beta^2\Phi \end{pmatrix} \tag{96}$$

In order to evaluate the Baumgarte parameters one can use the Taylor expansion up to order two for the constraint equation:

$$\Phi(t+h) = \Phi(t) + \dot{\Phi}(t)h + \ddot{\Phi}(t)\frac{h^2}{2} \tag{97}$$

where h is the time step. Considering that Φ is null at instants $t+h$ the latter can be rewritten as:

$$\ddot{\Phi}(t) + \frac{2}{h}\dot{\Phi}(t) + \frac{2}{h}\Phi(t) = 0 \tag{98}$$

By comparison with the relation for Baumgarte parameters we obtain:

$$\alpha = \frac{1}{h} \tag{99}$$

$$\beta = \frac{\sqrt{2}}{h} \tag{100}$$

The proposed approach is very simple and easy to implement, but it can lead to a series of numerical issues resulting in incorrect results. One should use another approach, e.g. the Euler integration as well as an analysis in the digital domain.

5.2 Relaxed Planar Complementarity Conditions

The relaxed planar complementarity conditions move from the *relaxed complementarity condition*:

$$h({}_ip)n({}_ip)_i^T f \leq \epsilon \tag{101}$$

Once the contact is made, the above condition cannot prevent the contact point to move on the walking plane, they are still free to move on the walking surface. This problem is often faced by imposing that the product between the planar velocities and tangential forces is equal to zero, the relaxed version of which writes:

$$-\epsilon_p \leq diag(t({}_ip)_i^T \dot{p})t({}_ip)_i^T f \leq \epsilon_p \tag{102}$$

with $\epsilon_p \in \mathcal{R}^2, \epsilon_p \geq 0$