Stable walking trajectory planning with multiple contacts using Gravito-Inertial Wrench Cone

Mohab Atanasious 1861248
Antonio Bozza 1836119
Saverio Taliani 1841465

April 24, 2023

Abstract

The following paper describes the main points of planning a feasible trajectory, for the Center of Mass (CoM) of a humanoid robot, using only the geometrical characteristics of the contact surfaces involved in the movement. In particular, it is analyzed the construction of the Gravito Inertial Wrench Cone, a 6D object that can give information on the feasibility of a CoM dynamical condition, by checking if the 6D vector of the total forces and torques exerted on the body from the contact surfaces (i.e. the current wrench) is in this 6D Cone. By exploiting this general constraint, it is then presented the construction of feasible CoM trajectories with heterogeneous combinations of contacts for different walking tasks, to be taken as reference for the real humanoid CoM movement.

Contents

1 Introduction

Humanoid robots are a class of robots that are designed to imitate human-like movements and perform tasks in environments that are specifically designed for humans. These robots are inherently complex systems to deal with, particularly when it comes to motion planning. In order to move, legged robots have to continuously change their contact with the environment, this means that they must carefully manage their force exchange with the ground and surrounding objects in order to maintain balance and avoid falling.
When a link of the robot is in contact with the ground, a distribution of friction and reaction forces is generated on the contact surface. Thankfully, when the contact area is convex, it is possible to consider only the forces exerted to the vertex of the support polygon.
As in many cases to plan the robot motions we refer to the position of the center of mass (CoM) and constrain resultant accelerations and torques applied to it. In this particular case, in order to plan motions of the Humanoid robot, we first plan a sequence of contacts that the robot will undergo and then we solve an optimization problem to obtain feasible trajectories for the robot's CoM that brings to a dynamically stable walking for the whole body.
In the following section, we present in detail how such constraints are generated starting from the friction cones of each contact point and how they are included in the final optimization problem.

2 Gravito-Inertial Wrench Cone

Under the assumption that torque limits are sufficiently large, the stability of the robot's contacts depends only on the gravito-inertial wrench, a six-dimensional object that represents the combined effect of gravity and inertia on the robot's motion. This wrench should lie within a convex cone called the Gravito-Inertial Wrench Cone (GIWC), which is generated by the contact constraints [1]. The GIWC provides a "universal stability criterion" for ensuring the robot's stability during movement, as it represents the set of all possible gravito-inertial wrenches that can be achieved while maintaining stability.

By constraining the robot's motion to remain within the GIWC, it is possible to plan stable and efficient trajectories for the robot's CoM that ensure it remains dynamically stable throughout the entire walking process. This approach to stability analysis is a critical concept in the analysis and planning of humanoid robot motion, as it provides a means of ensuring the robot remains stable and safe while performing complex tasks in a variety of environments.

2.1 Friction Cones and Surface Wrenches

As already mentioned in the introduction when a link is in contact with the environment we have a surface in which each point contributes to generate a friction and a reaction force.
We assume as it usually happens that each contact surface is a convex polygon. We want to prevent the robot to slip on the contact surface, hence we enforce static friction model to each contact point. We consider a contact force f to be *valid* when it is such that:

$$\begin{aligned} &\|f^t\| \leq f^n \mu_s \\ &f^n \geq 0 \end{aligned} \tag{1}$$

where μ is the friction coefficient, f^n and f^t denote respectively the normal and tangential components of the force with respect to a reference frame attached to the contact surface. In the following development, we will use the polyhedral approximation of the latter inequality, i.e.,

$$|f_x| \leq \mu f^n, \quad |f_y| \leq \mu f^n \tag{2}$$

where $\boldsymbol{f}^t = (f_x, f_y)$

Each cone has two possible representations the face and the span form.
The face form of a cone is given by the vectors which indicates a face of the cone by it's normal vector. The span form is instead represented by vectors which spanned generate the same space.
In the following analysis it is fudamental the ability to compute a form from the others since it will be important to exploit their advantages.

The reaction forces and torques given by the contact can be represented by a wrench $w = \begin{bmatrix} f \\ \tau \end{bmatrix}$ that is the result of contact forces at polygon's vertices.
A contact wrench is defined *valid* when it is generated by a set of *valid* contact forces.

To generate a *valid* contact wrench for each surface we have to consider constraints given by friction cones at each contact point and we sum their effects on the center of the surface generating a convex cone which defines the region containing acceptable wrenches. This 6D cone can be generated for each contact surface, as described in [2], and we implemented the construction of such surface cone in two different ways.
First of all, for each vertex of the contact 2D polygon we compute the friction cone, as the cone that has the tangent of the angle between itself and the contact surface equal to μ, friction coefficient. Through the MATLAB *cylinder* function we generate a set of points that belongs to the pyramidal cone approximation. These are 3D cones that we can express in span form, given that from the cone points that we found, we can write the four vectors that span the approximated cone (a square pyramid).

The surface cone is computed with respect to its center p, that is considered as the mean in the three space coordinate of the contact points. For a general set of contact forces applied to the vertices of the polygon, the resultant force for the surface is a direct sum of contact forces while the total torque is computed as $\tau = r_i \times f$ where r_i is $r_i = p_i - p$. This summation procedure is linear and can be encoded in A_{surf}:

$$A_{surf} = \begin{bmatrix} I_{3\times3} & \dots & I_{3\times3} & \dots \\ S(r_1) & \dots & S(r_i) & \dots \end{bmatrix} \tag{3}$$

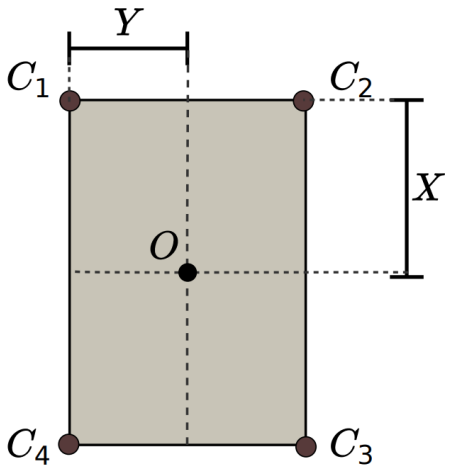
where $S(r_i)$ is the skew symmetric matrix of the vector r_i.
To get the surface cone we applied this linear mapping to each generating vector of the friction cones, multiplying A_{surf} by the stack of the span vectors:

$$V_{surf} = A_{surf} \begin{bmatrix} V^1_{point} \\ \dots \\ V^i_{point} \\ \dots \end{bmatrix} \tag{4}$$

A second approach is to consider the method described in [1], in which an algorithm for the computation of the Face form of a rectangular surface is presented. As a matter of fact the face form of each surface is found as:

$$\boldsymbol{U_{surf}} = \begin{bmatrix} -1 & 0 & -\mu & 0 & 0 & 0 \\ +1 & 0 & -\mu & 0 & 0 & 0 \\ 0 & -1 & -\mu & 0 & 0 & 0 \\ 0 & +1 & -\mu & 0 & 0 & 0 \\ 0 & 0 & -Y & -1 & 0 & 0 \\ 0 & 0 & -Y & +1 & 0 & 0 \\ 0 & 0 & -X & 0 & -1 & 0 \\ 0 & 0 & -X & 0 & +1 & 0 \\ -Y & -X & -(X+Y)\mu & +\mu & +\mu & -1 \\ -Y & +X & -(X+Y)\mu & +\mu & -\mu & -1 \\ +Y & -X & -(X+Y)\mu & -\mu & +\mu & -1 \\ +Y & +X & -(X+Y)\mu & -\mu & -\mu & -1 \\ +Y & +X & -(X+Y)\mu & +\mu & +\mu & +1 \\ +Y & -X & -(X+Y)\mu & +\mu & -\mu & +1 \\ -Y & +X & -(X+Y)\mu & -\mu & +\mu & +1 \\ -Y & -X & -(X+Y)\mu & -\mu & -\mu & +1 \end{bmatrix}$$

(a) Face form matrix



(b) Contact polygon

In this case the span form of the resulting cone is computed by deploying the *cdd* library double description conversion, as it is done in the analyzed paper simulations [2].

2.2 Gravito-Inertial Wrench Cone

Once the wrench cone of each contact surface is computed we need to join the given constraints in order to characterize the admissible forces and torques for the CoM. The arising cone is computed for each stance (a stance is a particular configuration of contacts), with the following procedure.

$$A_{stance} = \begin{bmatrix} -R_1 & 0_{3\times3} & \dots & -R_i & 0_{3\times3} & \dots \\ -R_1 \cdot S(CoP_1) & -R_1 & \dots & -R_i \cdot S(CoP_i) & -R_i & \dots \end{bmatrix} \tag{5}$$

This relation allows us to bring V^i_{surf} expressed in the $i-th$ surface frame directly to the CoM expressed in the laboratory frame.

$$V_{GIWC} = A_{stance} \begin{bmatrix} V^1_{surf} & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 \\ 0 & 0 & V^i_{surf} & 0 \\ 0 & 0 & 0 & \ddots \end{bmatrix} \tag{6}$$

In order to have a more suitable representation of the cone that allows to directly verify whether the Gravito-Inertial wrench of the robot belongs or not to it's feasibility region, we convert through the *cdd* library function, this span into a face form of the GIWC.

As described in [2], the computations that are operated each time that the stance (the contact surfaces) changes, are the computation of A_{stance} and V_{GIWC} with its face form U_{GIWC} to enforce in the optimal control problem the constraint on the current wrench:

$$U_{GIWC}\boldsymbol{w} \leq 0 \tag{7}$$

2.3 Optimization Problem

At this point we can set the Optimization problem to find a feasible trajectory for the CoM, imposing the constraint (7).

The code provided is a MATLAB script, called "traj_planning", which utilizes the CasADi library to solve the optimization problem related to robot locomotion. The goal of the optimization problem is to plan a walking trajectory for a humanoid robot from an initial position, with both feet on the floor, to a final point where the robot has to stop and balance statically.

To achieve this, the script begins by setting several parameters, such as:

- n_{stance}: desired number of stances or steps to take
- t_{stance}: desired time per step
- ss_{time}: time for single support
- ds_{time}: time for double support
- T: total simulation time
- dT: integration step
- m: mass of the robot
- g: gravity vector

These parameters are used throughout the script to define the problem and constraints.

Next, the script generates all of the possible contact surfaces for the robot's feet to interact with during walking. These surfaces are obtained through a function called "get_surf" and are stored in a variable called "surfaces".

After setting the parameters and generating the contact surfaces, the script begins to use the CasADi library to solve the optimization problem.

The first step is to define the state space and input as symbolic matrices. The state space consists of a vector of six elements $[\boldsymbol{p}^T_{\text{CoM}}\boldsymbol{v}^T_{\text{CoM}}]^T$, where $\boldsymbol{p}_{\text{CoM}} \in \mathbb{R}^3$ is a three-dimensional vector that represents

the position of the CoM, $\boldsymbol{v}_{\mathrm{CoM}} \in \mathbb{R}^3$ represents its velocity. The input is a vector of three elements, $\boldsymbol{u} \in \mathbb{R}^3$, representing the acceleration applied to the robot.

The robot dynamics are then defined using a continuous function called "continuous_dynamics", which represents the equations of motion for the robot. In this case, the evolution of the state is represented by a simple double integrator (8):

$$\begin{bmatrix} \dot{\boldsymbol{p}}_{\mathrm{CoM}} \\ \dot{\boldsymbol{v}}_{\mathrm{CoM}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_{\mathrm{CoM}} \\ \boldsymbol{u} \end{bmatrix} \tag{8}$$

The dynamics are then discretized using an explicit Euler method and a function called "discrete_dynamics". This function takes the state and input at the current time step and returns the state at the next time step.

The next step is to define the Gravito-Inertial wrench cone, which is a constraint on the forces and torques that the robot CoM is able to exchange while maintaining contact stability. This is achieved through a function called "gravito_inertial_wrench", which takes as input the contact surfaces and the center of mass of the robot and returns the Face form of the Gravito-Inertial wrench cone. Moreover, this function exploits the technique described in Section 2.1, using the algorithm for rectangular contact polygons and the *cdd* library for the Face/Span form switch.

On the other hand the current gravito-inertial wrench $\boldsymbol{w}_{\mathrm{GI}} \stackrel{\mathrm{def}}{=} (\ \boldsymbol{f}_{\mathrm{GI}}, \boldsymbol{\tau}_{\mathrm{GI}})$ of the robot, computed with respect to the origin of the laboratory frame, is defined by

$$\begin{aligned} \boldsymbol{f}_{\mathrm{GI}} &\stackrel{\mathrm{def}}{=} m\,(\boldsymbol{g} - \dot{\boldsymbol{v}}_{\mathrm{CoM}})\,, \\ \boldsymbol{\tau}_{\mathrm{GI}} &\stackrel{\mathrm{def}}{=} \boldsymbol{p}_{\mathrm{CoM}} \times m\,(\boldsymbol{g} - \dot{\boldsymbol{v}}_{\mathrm{CoM}}) \end{aligned} \tag{9}$$

The robot wrench at each time step is constrained to satisfy these equations.

Once the dynamics are defined, the optimization problem can be formulated. The objective function of the optimization problem is to minimize the sum of the squares of the input vector $\boldsymbol{u}$ at each time step i, which is defined as:

$$L = \sum_{i=1}^{T} \boldsymbol{u}_{\boldsymbol{i}}^{\boldsymbol{T}} \boldsymbol{u}_{\boldsymbol{i}} \tag{10}$$

For each time step, the script computes the Gravito-Inertial wrench cone using the current contact surface and center of mass of the robot, knowing that the stance for each time interval is determined by calling the "find_stance" function. The constraint on the wrench cone is then enforced using the "opti.subject_to" function:

$$U_{GIWC}\boldsymbol{w} \leq 0 \tag{11}$$

The dynamics of the robot are also enforced using this function, as well as a constraint that ensures the robot maintains a positive velocity in the walking direction.

Constraints are also imposed on the height of the robot's center of mass:

- the upper limit is $0.7\,m$ above the lowest contact surface;
- the lower limit is chosen to be about $0.3\,m$ below the upper bound value.

The initial and final positions of the robot are also imposed using "opti.subject_to" functions, as the final velocity that is constrained to be zero at the end of the task.

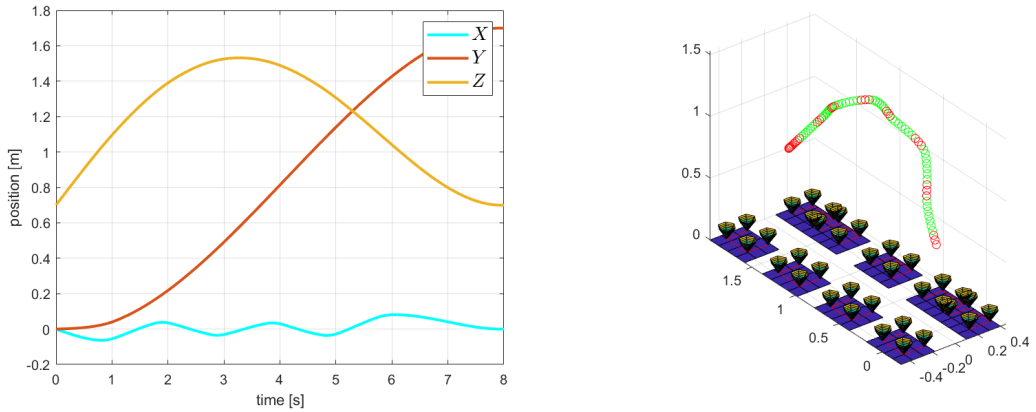
The overall optimization problem can be written as:

$$
\begin{aligned}
\min_{u} \quad & L(\boldsymbol{u}) \\
\text{s.t.} \quad & \dot{\boldsymbol{p}}_{\text{CoM}} = \boldsymbol{v}_{\text{CoM}} \\
& \dot{\boldsymbol{v}}_{\text{CoM}} = \boldsymbol{u} \\
& \boldsymbol{p}_{\text{CoM}}(0) = [0,\ 0,\ 0.7]^T \\
& \boldsymbol{p}_{\text{CoM}}(T) = p_{final} \\
& \boldsymbol{v}_{\text{CoM}}(T) = \mathbf{0} \\
& \boldsymbol{f} = m\left(\boldsymbol{g} - \dot{\boldsymbol{v}}_{\text{CoM}}\right), \\
& \boldsymbol{\tau} = \dot{\boldsymbol{v}}_{\text{CoM}} \times m\left(\boldsymbol{g} - \dot{\boldsymbol{v}}_{\text{CoM}}\right) \\
& U_{GIWC} \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{\tau} \end{bmatrix} \leq 0 \\
& z_{floor} + 0.4 \leq \boldsymbol{p}^{z}_{\text{CoM}} \leq z_{floor} + 0.7
\end{aligned}
\tag{12}
$$

where z_{floor} is the Z coordinate of the center of the lower current contact polygon in the stance.

Finally, the optimization problem is solved using the IPOPT solver, and the solution is obtained using the "opti.solve()" function. The solution provides a walking trajectory for the robot that satisfies all of the specified constraints.

Notice that the height constraint is necessary for the solver in order to give a proper behaviour to our robot. We show now the outcome of "flat surfaces" simulation without enforcing the aforementioned constraint.



(a) (b)

Figure 2: Cartesian position (a) and 3D plot (b), omitting the height constraint

Is glaring to see that the CoM height grows for the first 5 s to reach approximately a value of 1.6 m that is more than twice the nominal and desired height.
This behaviour has no physical meaning and brought us to think that constraining the CoM height to a certain desired region of the space was a reasonable solution. In the following section we will present trajectories generated properly constraining the CoM height.

3 Simulations

In this section we consider different scenarios which simulate various trajectories, represented by steps and contacts, to be accomplished by the robot. Different tasks are presented: the classic walking mode on flat surfaces, walking on oblique surfaces, walking on a sinusoidal shape surface, walking up the stairs, climbing on the wall and, finally, we consider the motion of getting above a box from an initial stable state on the ground.

The main point of these simulations is to highlight the possibility to find an optimal trajectory that satisfies the constraints described in the previous sections, in heterogeneous settings.

The physical main characteristic of the humanoid, seen as a point mass located at the CoM, are shown in the table 1.

Parameter	Value	Unit
t_{stance}	1	s
ss_{time}	0.7	s
ds_{time}	0.3	s
dT	0.1	s
m	50	Kg
g	-9.81	m/s^2
step_lenght	0.3	m
foot_lenght	0.24	m
foot_width	0.18	m
hand_lenght	0.18	m
hand_width	0.18	m
mu	0.65	-

Table 1: Humanoid characteristics

The n_{stance} value is variable through the simulations, but the total simulation time is derived as:

$$T = t_{stance} * n_{stance}/2 + ds_{time}$$

It is also evaluated the mean time, over 30 runs, required to find the optimal trajectory for each task, these data are presented in table 2 for each simulation in seconds. The machine considered for this evaluation is equipped with an i7-7700HQ and 16 Gb of RAM.

Task	Solver time
Flat straight walk	2.4328 s
Flat sinusoidal walk	20.2367 s
Oblique surface walk	8.3200 s
Sinusoidal ground walk	13.0002 s
Stairs walk	1.6921 s
Wall climbing	1.3671 s
Box climbing	3.9502 s

Table 2: The solver time to find the optimal trajectory for each task, all times are reported in seconds

3.1 Flat surfaces walk

As a first case, we considered a straight walk, which is also the simplest one for our aim. We want the point mass to go from the position $[0, 0, 0.7]$ to $[0, 1.7, 0.7]$, a straight line along the y axis, with a step length of $0.3m$. The surfaces, along with the optimal trajectory, can be seen in figure 3. The movement evolution starts from a double support situation and ends in the same condition, giving the constraint for the CoM to have a zero initial and final velocity, in every direction.

The obtained trajectory has a sinusoidal shape, in the single support condition, in green dots, it approaches the support surface, while in double support, in red dots, it tends to the center. This is clearly an effect of the GIWC constraints from the stances generated in the walk task.

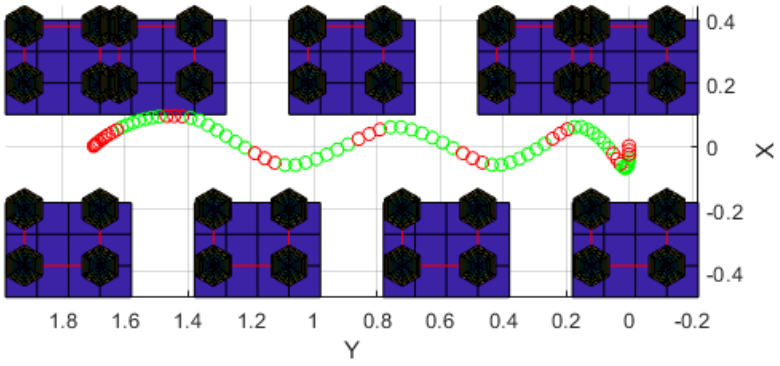
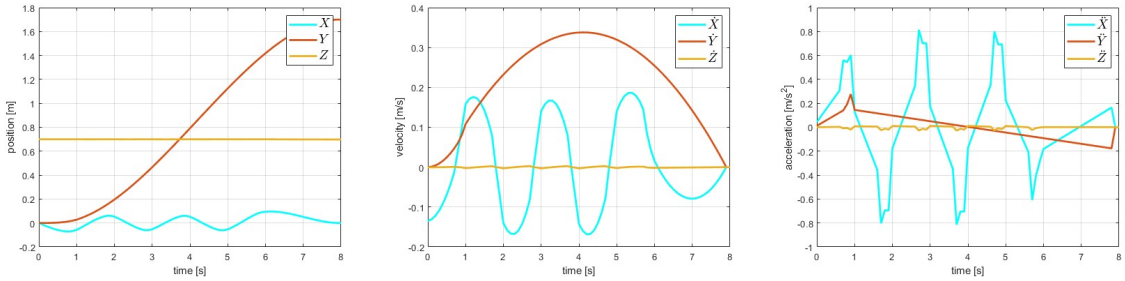


Figure 3: Trajectory in 3D of the CoM in the planar contact surfaces case

We can analyze more in detail this trajectory in figure 4, where the evolution of the cartesian position, velocity and acceleration of the CoM is visualized.



(a) (b) (c)

Figure 4: Cartesian position (a), velocities (b) and accelerations (c) of the horizontal straight walk task

It is also considered a similar setting: a planar walk with a sinusoidal shape in the x-y plane, imagining the presence of two obstacles to avoid, that make a straight walk not possible. The evolution is shown, as for the other planar task, in figures 5 and 10.

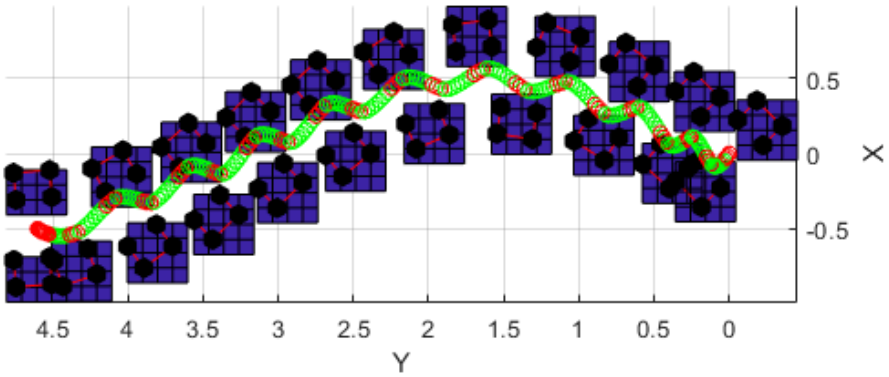
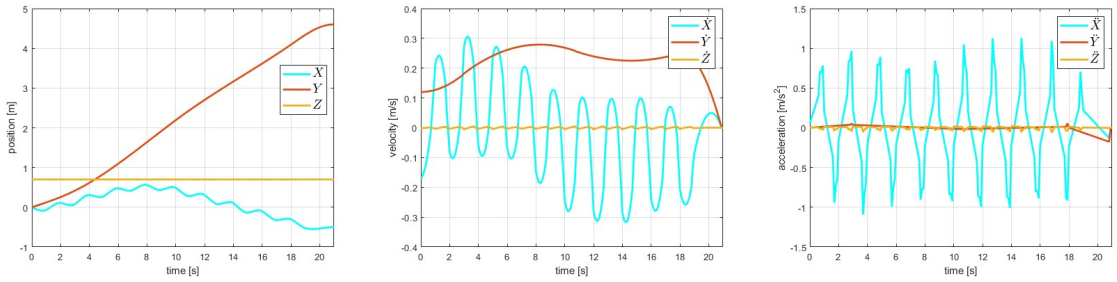


Figure 5: Trajectory in 3D of the CoM in the planar sinusoidal contact surfaces case

As shown in the figures, the task is accomplished as for the straight one, oscillating between the contact surfaces during the walk, and finishing with a zero velocity condition. The overall position,



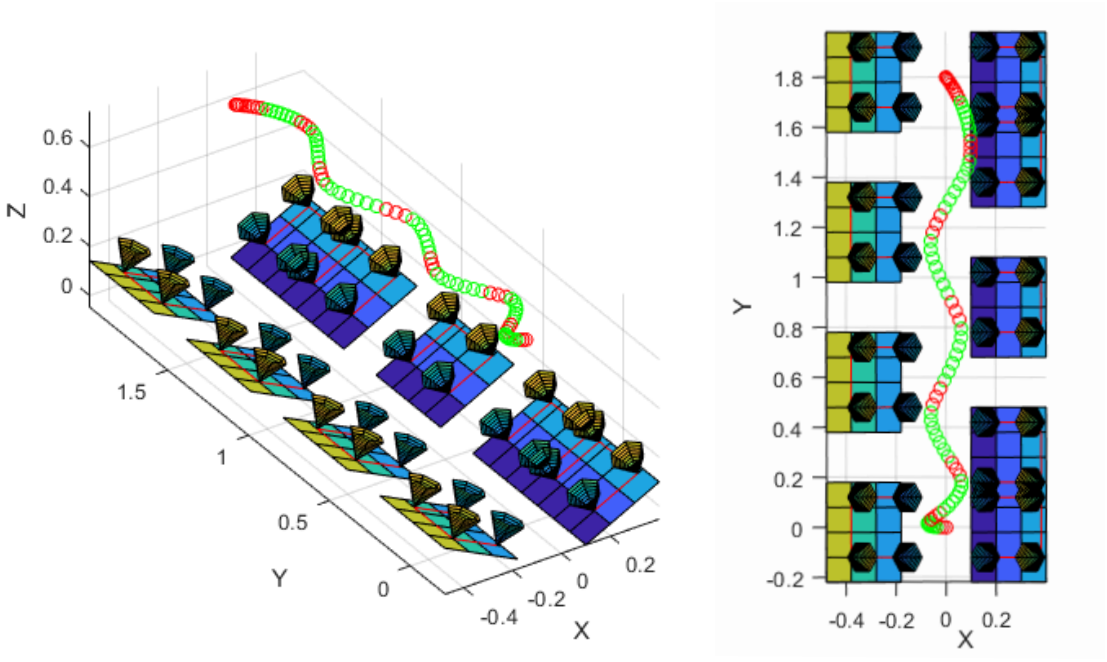
(a) (b) (c)

Figure 6: Cartesian position (a), velocities (b) and accelerations (c) of the sinusoidal flat walk task

velocity and acceleration profile have the same shape as in the previous simulation, with the addition of a significant change in the X coordinate due to the shape of the surface's path.

3.2 Oblique surface walk

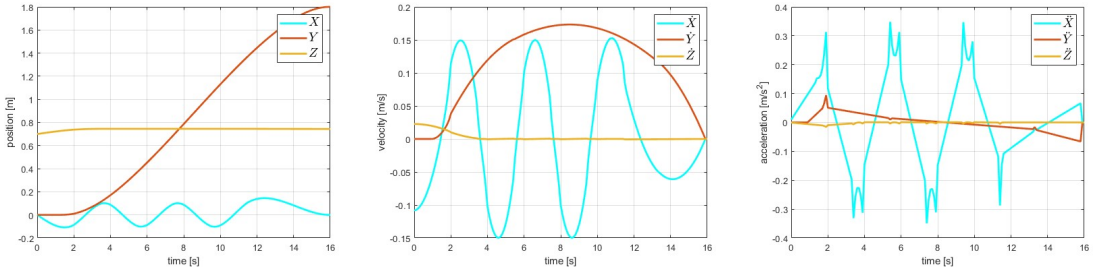
In the second simulation, we considered another task in which the contact surfaces are not flat and, as a consequence, the reaction forces from the surface will affect the stability of the posture if the force are not well-calculated. The task is similar to the previous one, and has the same characteristics except for the orientation of the contact surfaces, the goal (the last configuration of the CoM) is the same and the contact surfaces are positioned like the flat case; in particular, the length of the steps is the same as in the previous task which is 30 cm. We consider a time of 1.4 seconds for the single support situation and 0.6 seconds for the double support. A particular consideration is the addition of 2 seconds at the end to let the CoM assume a zero velocity at the last step. Also in this case, as we can expect, the motion of the CoM will have a shape similar to a sinusoidal one with respect to the Y axis making the CoM nearer to the contact surface(s). The generated trajectory is described in figure 7 where it is provided a 3D visualization: the red circles represent the position of the CoM in the double support case and the green ones represent the single support case.



(a) (b)

Figure 7: Trajectory in 3D of the CoM in the oblique contact surfaces situation

On the other hand the position, velocity and acceleration profiles are described in figures 8a, 8b and 8c, respectively. We obtain very smooth position trajectories: at the beginning the CoM evolves only with respect to the X axis to bring it nearer to the support surface of the next stance. This motion is predictable and expected due to the intuition and similarity we obtain by our experience in walking. The advantage is the reduction of the computation time needed to solve the optimization problem and to obtain the needed motion of the CoM; the computational time is 8.3 seconds. The height of the CoM does not have meaningful changes as it stays at the height of 75 cm.



(a) (b) (c)

Figure 8: Cartesian position (a), velocities (b) and accelerations (c) of the oblique straight walk task

On the other hand, the velocity profiles are smooth with a zero value at the end to keep the robot in a stable static configuration. Finally, the acceleration profiles are periodic and we do not expect them to be smooth as they are the inputs and could be discontinuous.

3.3 Sinusoidal ground walk

We have considered also a particular ground condition, in which we have a variable contact surface orientation with respect to the world frame. It is presented in this subsection a task for a straight walk evolution along a sinusoidal wave, that in this case evolve on the z axis and is function of the y coordinate.

The equation of this wave is the following:

$$z = 0.4 * (cos(2 * \pi * y/3.6)) - 0.4; \tag{13}$$

The surfaces in this case are taken as in the first simulation of a stright planar walk, however in this case the z-coordinates for the contact polygons vertices are chosen to make the shape lie on the plane tangent to the wave at the center of the foot.

Follows the 3D surfaces considered for the motion along with the optimal trajectory obtained, in figure 9.

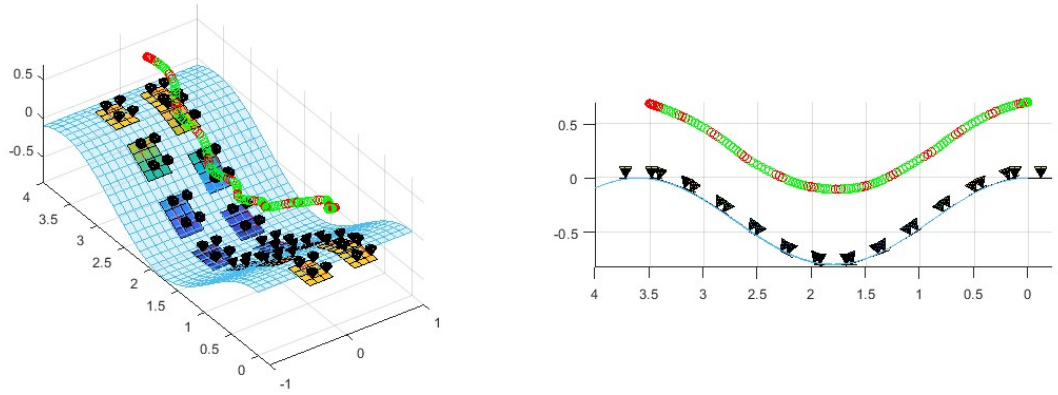
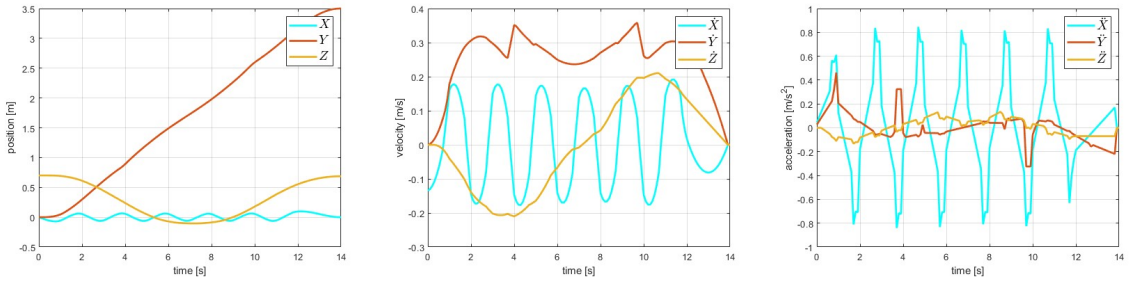


Figure 9: Trajectory in 3D of the CoM in the sinusoidal ground contact surfaces situation



(a) (b) (c)

Figure 10: Cartesian position (a), velocities (b) and accelerations (c) of the sinusoidal ground walk task

The position, velocity and acceleration profiles show us an evolution of the CoM that is consistent with the previous results, from the planar walk simulations, with in addition the changes in the z coordinate to follow the sinusoidal surface due to the bounds in its height.

3.4 Stairs walk

An important problem to solve for humanoid robot is to give them the ability to walk on surfaces that are non co-planar. Classic examples to underline the importance of this ability are stairs.
In this task our robot is requested to perform steps of $0.3m$ along the y axis and $0.1m$ along the z axis. Posing the initial position to be $[0, 0, 0.7]$ and posed as target position $[0, 1.7, 1.3]$.
The computed trajectory is shown in figure 11.

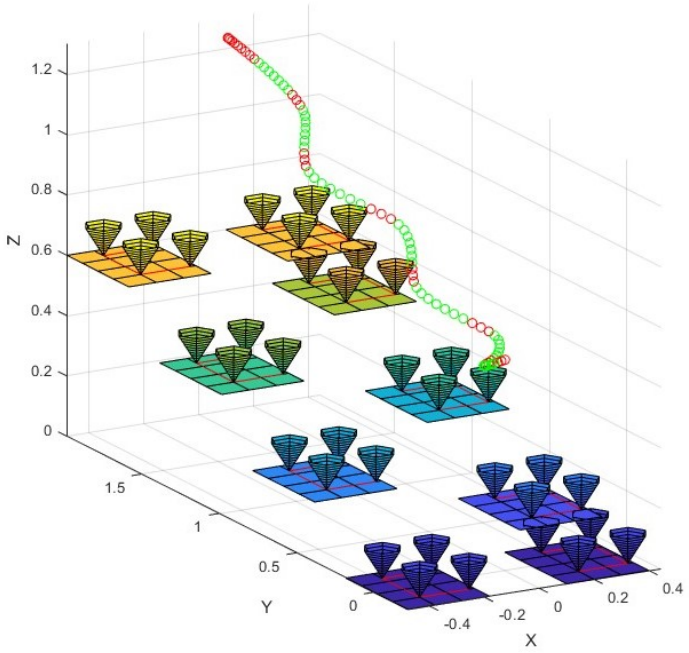
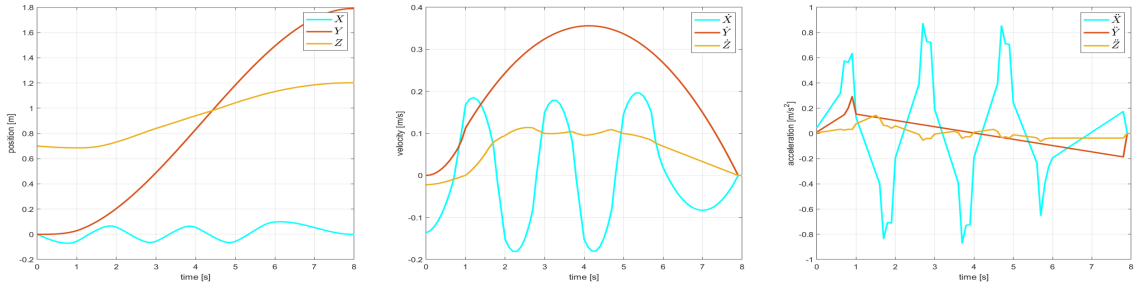


Figure 11: Trajectory in 3D of the CoM on stairs

We can see that the CoM increases its height almost constantly while to satisfy constraints of GIW cones the CoM oscillates around the x axis. One thing to notice is that since we forced the robot to have zero velocity at the end we have a different behaviour in the last step. The CoM seems to deviate from x axis more than previous steps, visually we can see that its projection on the ground falls over

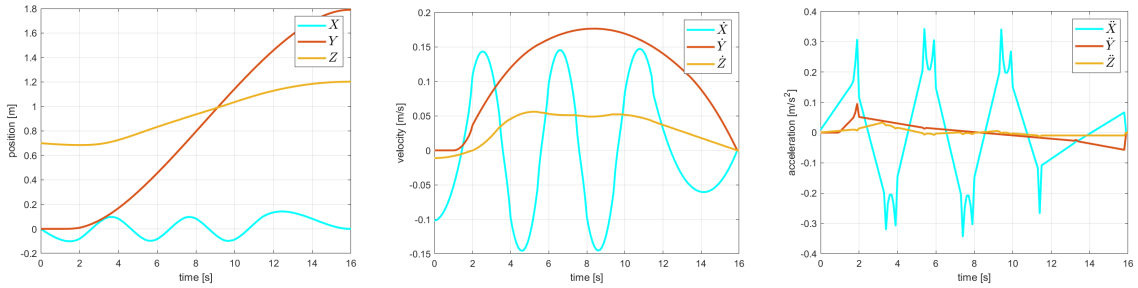
the support area. This behaviour is comprehensible, to stop the robot the velocity has to be reduced, so the CoM has to stay nearer to the support area to maintain stability.
We can visualize better the result analyzing the evolution of position, velocity and acceleration in time. As we can see in 12a the goal is reached correctly the center of mass height grows smoothly while the



(a) (b) (c)

Figure 12: Cartesian position (a), velocities (b) and accelerations (c) of the stairs walk task with 1s per step

robot is going up the stairs. As mentioned before at $6s$ the x position varies to perform the last step, but it varies more than in previous steps, this can be interpreted also looking at the correspondent acceleration. Probably in order to stop correctly the optimizer starts to reduce accelerations and velocities and this bring the necessity to put the CoM of the robot nearer to the contact area in order to satisfy the GIWC constraint. The maximum velocity required (12b) is $0.4\,m/s$ in the walking direction, while on x axis it varies sinusoidally to let the robot maintain feasible contacts.
By increasing the time per step up to $2s$ we obtain similar results. The maximum velocity required



(a) (b) (c)

Figure 13: Cartesian position (a), velocities (b) and accelerations (c) of the stairs walk task with 2s per step

(13b) is clearly less than before and the overall behaviour is a reasonable scale of the previous analysis. A slightly difference is encountered in the x coordinate in which the CoM position is subject to a more pronounced oscillation.

3.5 Climbing on the wall

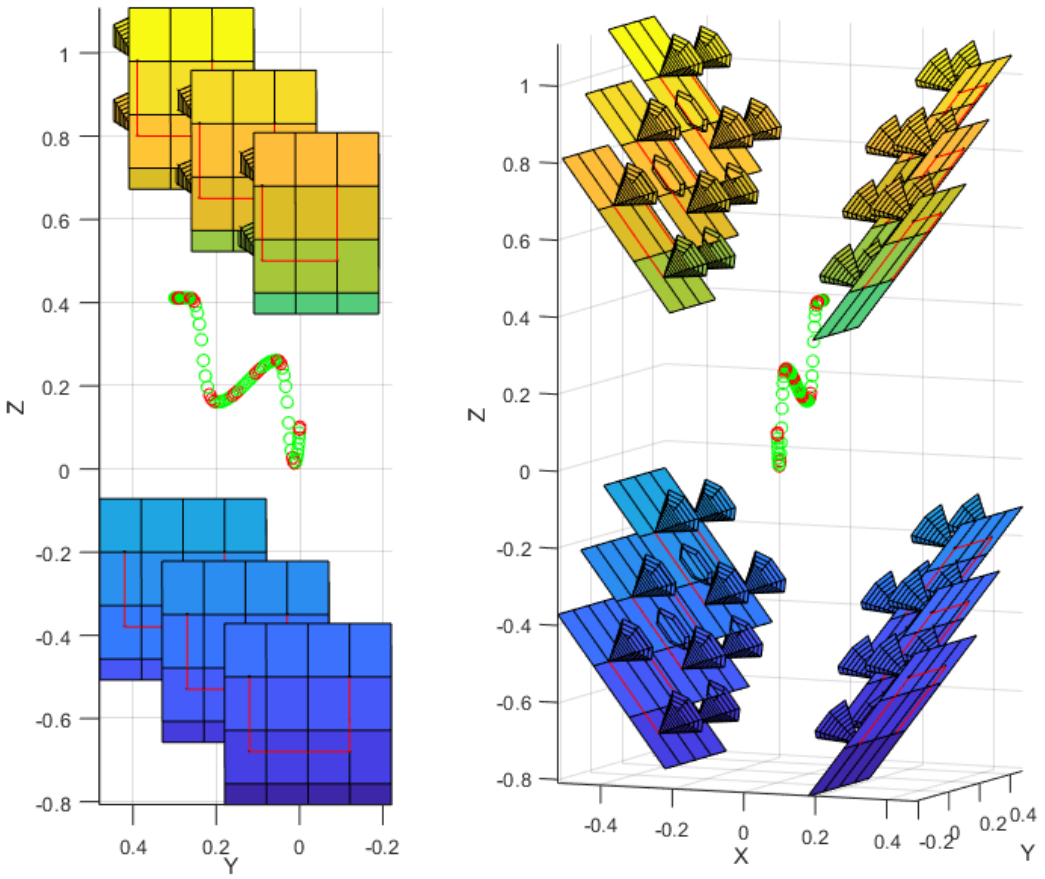
Another task which seemed to be interesting is the climbing on the wall. The robot has to use both legs and hands to go up using contact surfaces on two facing walls.

In this simulation, there was an implementation limitation relative to the use of the libraries: a vertical surface can not be used and we had to consider a set of surfaces inclined by 60° with respect to the X axis. The robot is given in a situation where it holds itself by the legs and the hands orienting them to the walls, it releases one contact at the time to establish a new one right above and displaced along the Y axis, this operation is repeated 2 times for all the contact surfaces. In addition, for this specific task, the constraint on the Z positional component is stricter for the lower bound:

$$z_{floor} + 0.6 \leq \boldsymbol{p}^{z}_{\text{CoM}} \leq z_{floor} + 0.7$$

This is done to have smaller distances between the CoM and the upper contacts compared with the less restrictive constraint case.

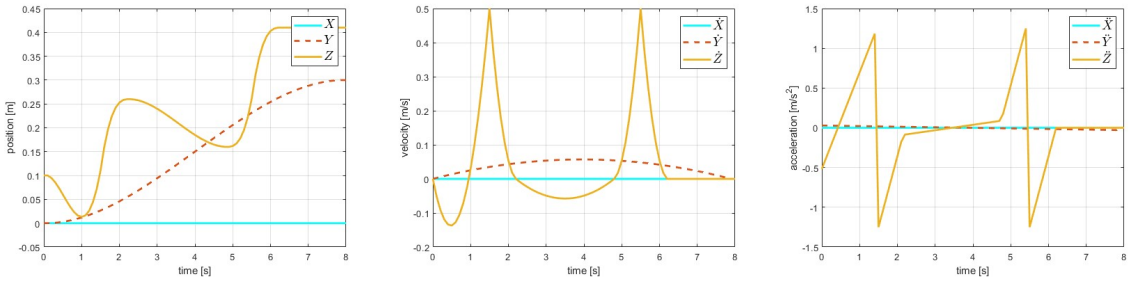
The 3D trajectory is described in figure 14 and the position, velocity and acceleration profiles are described in figures 15a, 15b and 15c, respectively.



(a) (b)

Figure 14: Trajectory in 3D of the CoM for the climbing task

As expected, there is no motion with respect to the X axis due to the fact that being in the middle is a stable configuration and there is no need to have any displacement between the 2 walls. On the other hand, by having a cost function as the quadratic form of the acceleration, it is a minimum to not have any movement in the X axis. In this simulation the upper and lower bounds imposed to the height of CoM are relevant; as a matter of fact, the optimization problem makes use of this constraint to minimize the cost function avoiding a very high peak in the acceleration as the values of the height



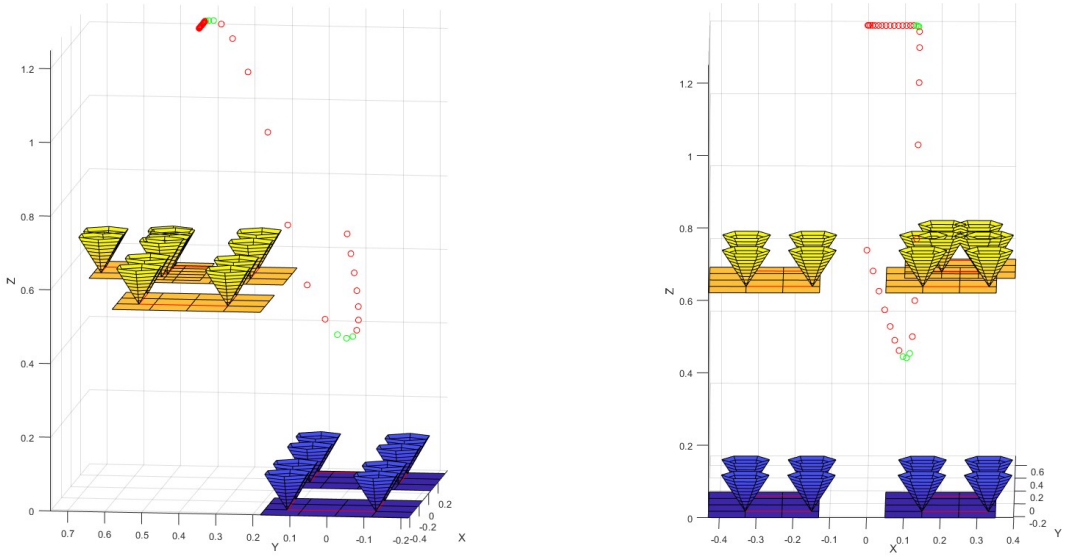
(a) (b) (c)

Figure 15: Cartesian position (a), velocities (b) and accelerations (c) of the climbing task

changes with the change of the stance. To avoid this, we note that the height of the CoM oscillates and increases to reach the goal allowing the presence of the needed velocity and minimizing the cost function. The robot accomplish the task successfully in 8 seconds and it took 1.6 seconds to find this optimal solution.

3.6 Box

In conclusion, we wanted to try an interesting setting: we thought the robot standing in front of a box with the goal to step over using an hand to help it self in the operation. We set a box of $0.55m$ posed at a distance of $0.1m$ from its static position. The center of mass initially lowers down in practice the

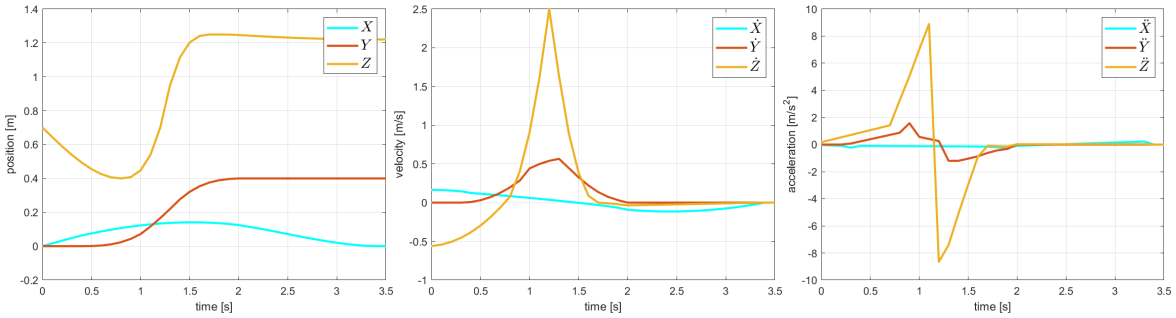


(a) (b)

Figure 16: Trajectory in 3D of the CoM for the box task

robot is leaning forward to put the and on the box. Then the CoM is quickly brought upward to jump on the box and at the end it goes to the desired position with zero velocity.

As can be noticed in this case the acceleration and velocity required are quite large during the jump, anyway the position profiles are smooth and reasonable. Moreover, with the final condition of zero velocity a solution is found only thanks to the presence of the hand contact in front of the feets, that allows forces and torques to decelerate when on the box.



(a) (b) (c)

Figure 17: Cartesian position (a), velocities (b) and accelerations (c) of the Box task

4 Conclusions

As a result of this wide range of simulations, we have shown how the Gravito-Inertial Wrench Cone can be easily and effectively used to plan feasible trajectories for a humanoid robot.
This methodology gives huge advantages, in order to plan a series of contacts it is sufficient to know the friction coefficient and the desired position of the contact points; the constraint can be computed only once per stance and can include any number of contact surfaces without changing the expression. Finally impose the GIWC it is sufficiently robust to parameter variations.
At last to verify the belonging of a wrench to the cone it is very simple and straightforward.

However some drawbacks remain: the switch from face to span form gives, with the current implementation, numerical problems if the surfaces are rotated more than $\pi/3\,rad$ (in absolute value) with respect to the X or Y axis, and the same numerical problems are present also if the surfaces in a stance have an orientation that differs by an angle in the order of $10^{-2}rad$. The possibility to perform the computation remains by increasing the numerical resolution, but drastically sacrificing the computational time. Moreover we cannot compute the polyhedron for the GIWC with vertical surfaces.

As a last consideration it is difficult to represent the GIWC, so it is not possible to inspect more detailed information about the belonging of the Gravito-Inertial wrench to the feasibility cone. It would be interesting to compute a measure of distance between a feasible wrench and the border of the polyhedron in order to add regularization terms in the cost function to center as much as possible the vector in the stability region.

References

[1] Stéphane Caron, Quang-Cuong Pham, and Yoshihiko Nakamura. Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5107–5112, 2015.

[2] Stephane Caron, Quang Cuong Pham, and Yoshihiko Nakamura. Leveraging cone double description for multi-contact stability of humanoids with applications to statics and dynamics. 07 2015.