Humanoid walking with TSID*

*Assignement n.1 of the course of *Optimization-Based Robot Control* held by prof. Del Prete Andrea, phd. Grandesso Gianluigi

Dalle Vedove Matteo
matteo.dallevedove@studenti.unitn.it
ID. 232604

Rizzardi Alessandro
alessandro.rizzardi@studenti.unitn.it
ID. 233221

I. The Python code

The fist thing was to figure out how the given code works. After a look at the file *hw1_LIPM_to_TSID_template.py*, we understood that the function *compute_3rd_order_poly_traj* is actually called in two parts: one to interpolate the trajectories of each step in the horizontal plane (coordinates x and y) and one for the vertical direction (z). The latter step is divided in two sub-parts: the first to gain foot height, the second to return to ground.

Anyway, in each function call only the initial $x(0) = x_i$ and final $x(T) = x_f$ point of the trajectories are given, but it is require to perform a cubic interpolation

$$x(t) = a + bt + ct^2 + dt^3 \tag{1}$$

To fully determine the unknown coefficient a, b, c, d 4 boundary conditions are required, but only 2 are given. For this reason we assumed that the trajectories must start and approach the final configuration with a zero velocity ($x'(0) = x'(T) = 0$). This gives the solution

$$a = x_i \quad b = 0 \quad c = 3\frac{x_f - x_i}{T^2} \quad d = -2\frac{x_f - x_i}{T^3} \tag{2}$$

Based on the solution (2), the python code has been developed: firstly we initialized the vector (actually matrices) of the state $x(t)$ and it's 2 first derivatives $x'(t), x''(t)$ with a number of columns $\lceil T/dt \rceil$; once the coefficients has been determined, we iteratively assign the value of the coordinate at each time-step.

II. Questions

A. Weight tuning

The first challenge is to tune the weights of the task associated to: foot trajectory (*w_foot*), center of mass (COM) trajectory (*w_com*) and the posture (*w_posture*).

Preliminary it's important to understand how changing such values impact in the behaviour of the system. From a qualitative point of view it has been observed that:

- increasing too much *w_posture* leads to a still robot: indeed the best way to achieve a *correct* posture of the robot is by leaving it stationary (thus not following the desired motion). Conversely, leaving it too small wouldn't prevent the system to reach unstable configuration, leading to a robot falling;
- increasing too much *w_foot* helps at correctly tracking the foot trajectories, however this also introduces bad controls of other joints that, over time, leads to an unstable configuration.

Table I reports the key milestones that have been followed to achieve to the final result (reported later) and the will be discussed here in detail.

We firstly increase the weights of feet and COM of a couple of order until we obtained a complete working walk (*w_foot* = 10, *w_com* = 10). To achieve better feet trajectories we increased the corresponding weight and we discovered that the performances of the feet continue to improve until the value of 200 then the simulation fail, so we set the foot's trajectories weight to such value.
To obtain better performances with the COM, we increased its weight and we observed that it could have been increased of several order of magnitude without compromising the walk, but we find a reasonable weight *w_com* $= 2 \cdot 10^3$.
At the end we consider to not set the posture's weight to much higher for the reason previously explained and, after some trial&error on different magnitude orders (10^{-2}, 10^{-1},1), we settled to a value *w_posture* $= 1 \cdot 10^{-1}$ that was providing us the best performances.

After all the iterations, the chosen weights that have been used throughout the whole homework are the following:

$$\begin{cases} w\_com & = 1 \cdot 10^3 \\ w\_foot & = 2 \cdot 10^2 \\ w\_posture & = 1 \cdot 10^{-1} \end{cases} \tag{3}$$

Fig. 2 and 3 are reporting the results of the simulation with this tuned weights. Broadly speaking, the results obtained with the real trajectory of the COM compared with its reference are quite good; the position, fig. 2(a), in z axis shows a little oscillation but the amplitude is small consider the different order of the ordinate x and y. The little oscillation of the COM in z is visible also looking at velocity and acceleration

Table I: key milestones achieved while searching for the optimal weight.

w_com	*w_posture*	*w_foot*	Figure reference
$5 \cdot 10^{-1}$	$2 \cdot 10^{-2}$	$1 \cdot 10^{-1}$	
10	$2 \cdot 10^{-2}$	10	fig. 1
$1 \cdot 10^3$	$1 \cdot 10^{-1}$	$2 \cdot 10^2$	fig. 2, fig. 3

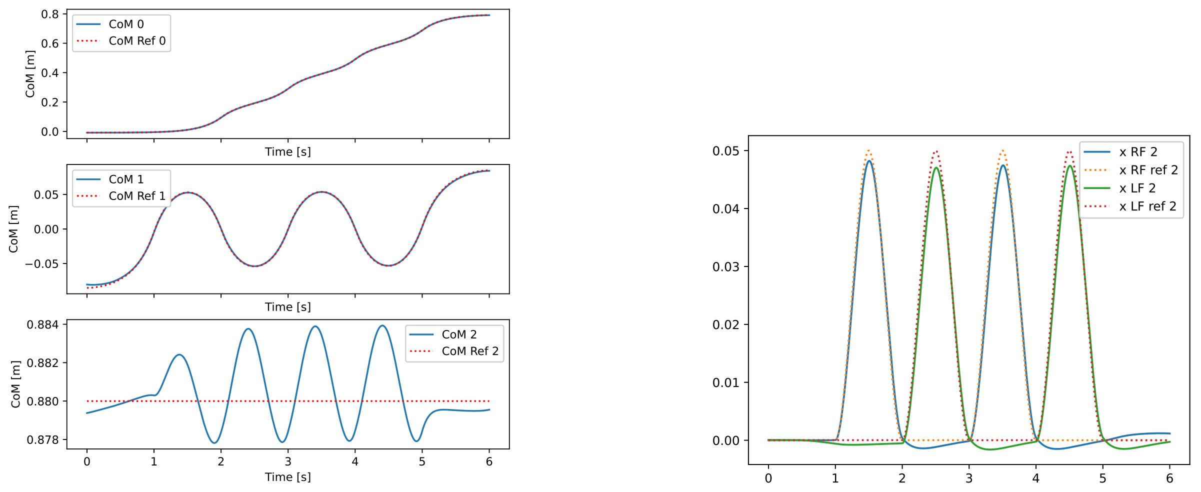

(a) (b)

Figure 1: Center of mass position (a) and foot trajectory along the z direction (b); parameters:*w_com*= 10, *w_posture* $= 2 \cdot 10^{-2}$, *w_foot* $= 10$

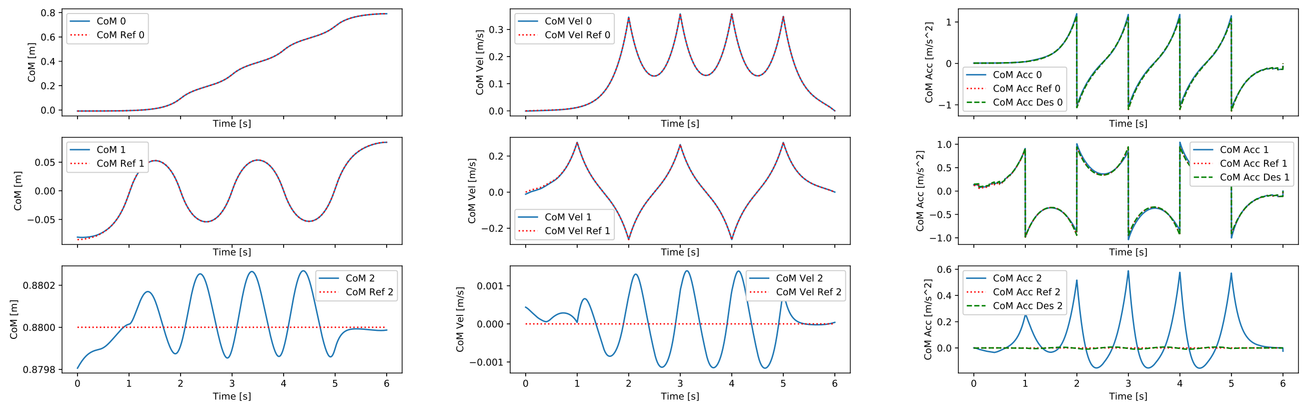

(a) (b) (c)

Figure 2: Center of mass position (a), velocity (b) and acceleration (c); parameters: *w_com*= $1 \cdot 10^3$, *w_posture* $= 1 \cdot 10^{-1}$, *w_foot* $= 2 \cdot 10^2$

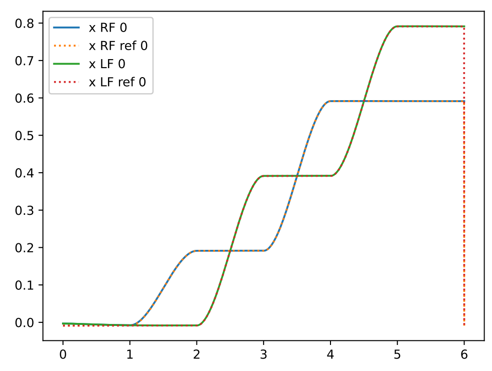
but, as already pointed, the values of these are small.
Fig. 3 shows the trajectory of the feet in x and z (y axis is quite trivial, it is just a straight line for both feet). In x coordinate the tracking us almost perfect, while in z there's present a little error when the foot should touch ground: the coordinate value becomes negative (still with very low values), but this is due to the fact that we are not perfectly simulating the presence of solid ground which will generate a constraint; the solver in facts just integrates the equations of motion of the robot who cannot intrinsically model the presence of ground.

B. *Squat walk*

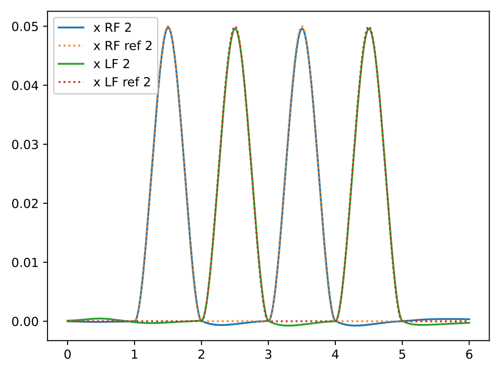
Given the parameter (3), we proceeded by enabling the squat task on the robot. The associated default weight *w_squat*= 10 appears to be too low: indeed squatting is a task that's conflicting with the trajectory of the center of mass. In particular why *w_com* is forcing all 3 components of the center of mass's trajectory, the squat only drives the z coordinate.

With this premise, in order to let the robot walk while squatting we decided to set the associated task weight a few order of magnitude higher then the other one:

$$w\_squat = 1 \cdot 10^6 \tag{4}$$



(a)



(b)

Figure 3: Feet trajectory for axis x (a) and z (b); parameters: $w\_com = 1 \cdot 10^3$,$w\_posture = 1 \cdot 10^{-1}$, $w\_foot = 2 \cdot 10^2$

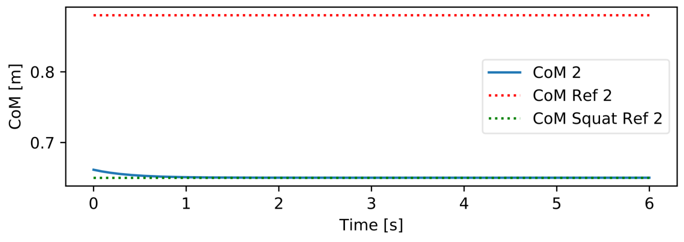


Figure 4: Squatted walk, trajectory of the center of mass on the vertical z component only; parameters: $w\_com = 1 \cdot 10^3$, $w\_foot = 2 \cdot 10^2$, $w\_posture = 1 \cdot 10^{-1}$, $w\_squat = 1 \cdot 10^6$, $kp\_squat = 1 \cdot 10^1$. Other components have not been reported as there's no noticeable difference from fig. 2(a).

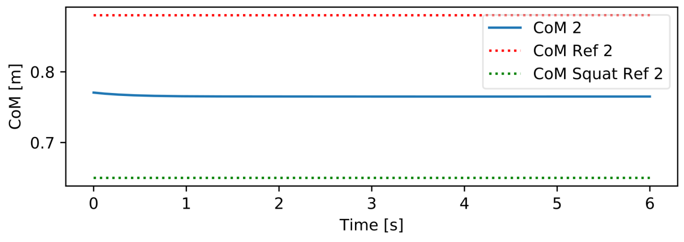


Figure 5: Squatted walk, trajectory of the center of mass on the vertical z component only; parameters: $w\_com = 1 \cdot 10^3$, $w\_foot = 2 \cdot 10^2$, $w\_posture = 1 \cdot 10^{-1}$, $w\_squat = 1 \cdot 10^3$, $kp\_squat = 1 \cdot 10^1$. Other components have not been reported as there's no noticeable difference from fig. 2(a).

As fig. 4 shows, the robot is able to correctly track the trajectory of the center of mass without running into singular configurations or falling; the result has been obtained using the default proportional gain $kp\_squat = 1 \cdot 10^1$.

For sake of completeness, we tried to lower the task's weight to be comparable to the one of the center of mass trajectory ($w\_squat = 1 \cdot 10^3$), but as expected the COM stays in between the 2 desired z trajectories, as shown in fig. 5.

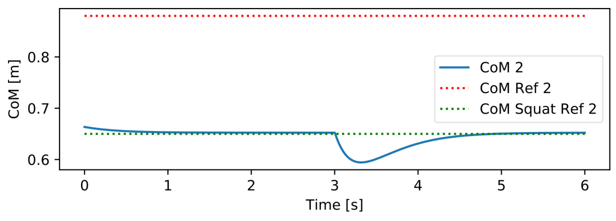
We also tried to increase the proportional gain to the value $kp\_squat = 1 \cdot 10^3$ (leaving the higher weight $w\_squat = 1 \cdot 10^6$): the result in terms of the COM tracking capability are identical to the one considering a lower proportional gain (see fig. 4). This however describes only partially the picture of the motion: by a qualitative point of view the result appears to be unstable in it's overall motion by having the torso swinging a lot, specially in the initial part of the trajectory. This is typical of a proportional control; for example increasing further the gain to $kp\_squat = 1 \cdot 10^4$ leads to a system that is too unstable in such a way that the simulator is not even good a simulating the initial time steps.

C. Upright walk with horizontal push

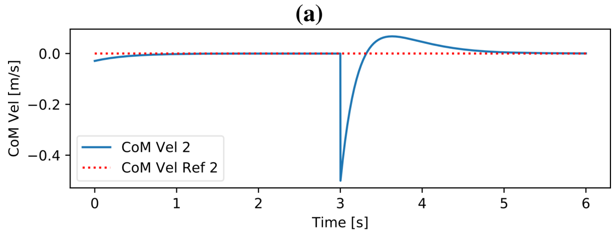
In the next step it is required to enable a lateral push at half walk to the COM. Running the simulation using the weights obtained in (3) the humanoid is able to complete the walk without falling.

In order to understand what are the main factor that allowed the robot to succeed in the task we started changing the previously impose weights until we were able to force the biped's fall. After some tries we understood that the weights of the COM (*w_com* in the configuration's script) was the main parameter which influenced the fall, indeed by setting a lower weight $w\_com = 2$ we managed to make the robot fall.

We also tried to play with the value of the proportional coefficient related to the PID control (*kp_push*). We observed that by increasing this value, the response to an error in the COM position would have helped the robot in countering the push, but we also noticed that such contribution of the proportional gain was very limited and moreover it introduced some instability in the walk(visible only watching the simulation not the graphs).

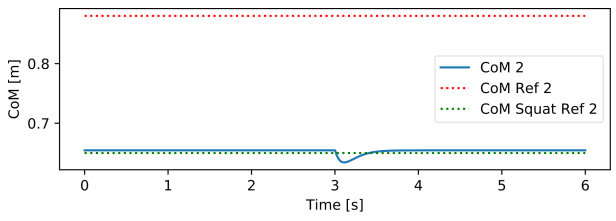


(a)

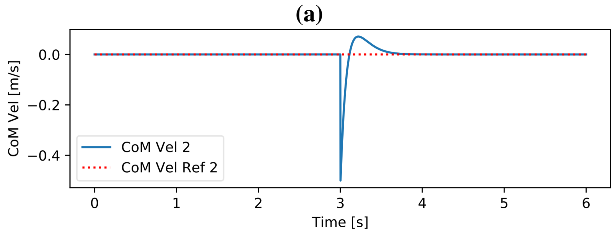


(b)

Figure 6: Squatted walk with vertical push, (a) trajectory of the center of mass and (b) associated velocity of the z component only; parameters: $w\_com = 1 \cdot 10^3$, $w\_foot = 2 \cdot 10^2$, $w\_posture = 1 \cdot 10^{-1}$, $w\_squat = 1 \cdot 10^5$, $kp\_squat = 1 \cdot 10^1$, $kp\_com = 1 \cdot 10^1$. Other components have not been reported as there's no noticeable difference from fig. 2.



(a)



(b)

Figure 7: Squatted walk with vertical push, (a) trajectory of the center of mass and (b) associated velocity of the z component only; parameters: $w\_com = 1 \cdot 10^3$, $w\_foot = 2 \cdot 10^2$, $w\_posture = 1 \cdot 10^{-1}$, $w\_squat = 1 \cdot 10^4$, $kp\_squat = 1 \cdot 10^2$, $kp\_com = 1 \cdot 10^1$. Other components have not been reported as there's no noticeable difference from fig. 2.

So in case the robot falls, the best solution is in general to increase the weight *w_com* related to the COM's task.

The main drawbacks in this strategy is that increasing the COM's weight too much it could becomes more difficult for the robot to succeed in others task related with the COM position, for example setting a very high *w_com* will make more difficult the squatting action that will need an higher value of the squat-task weight.

D. Squatted walk with vertical push

Opposite to the previous case, the robot is in this case subjected to a push acting on the vertical z axis that instantaneously changes the velocity by $0.5m/s$ toward ground direction.

Fig. 6 shows the simulation performed using the weights in (3), additionally with

$$\begin{cases} w\_squat & = 1 \cdot 10^5 \\ kp\_squat & = 1 \cdot 10^1 \\ kp\_com & = 1 \cdot 10^1 \end{cases} \tag{5}$$

Such configuration still allows the robot not to fall during the whole simulation; moreover we don't experience any "unstable" behaviour of the robot that's not swinging too much, even when subjected to the external load.

As suggested, we also increased the proportional gain of the squat to the value $kp\_squat = 1 \cdot 10^2$ while decreasing by a order of magnitude the associated gain ($w\_squat = 1 \cdot 10^4$). Focusing just on the trajectories along the vertical direction (no significant differences can be noticed in the x, y directions), fig. 7 seems to show an overall better result. As expected, increasing the proportional gain improves the convergence time of the center of mass to the desired configuration, passing from almost $2s$ to less then $1s$. This however comes with the drawback of a stiffer robot: what plots don't show, as already stated in the squatted walk, is that the robot tends to oscillate more while moving due to the stiffer behaviour, specially in the initial part of the motion.

Overall, we prefer the first combination of parameters (5) that provides a smoother walk of the whole robot, even still the convergence rate can be improved.

As a last note, we want to report a small issue concerning the foot trajectory along the vertical direction. As fig. 8 shows, when the push is applied to the robot, also the feet is consequently moved, reaching a negative coordinate that's physically not feasible (as the foot should go underground). This behaviour can be explained, as for the first question, by acknowledging that we are not using a proper simulator, but we are simply integrating the equations of motion of the robot (and in the problem formulation there's no a real constraint modelling the ground).

III. Implementation Notes

To simplify our trial&error phase, we slightly modified the file *hw1_conf.py*; in particular to easily jump from one question to another, we implemented a switch. In this way it's also easier to set different weights depending on the different tasks; still the final result is using similar values for both weights and gains.

Also *hw1_tsid_biped_walking.py* has been slightly modified; in particular we added few lines of code to program-

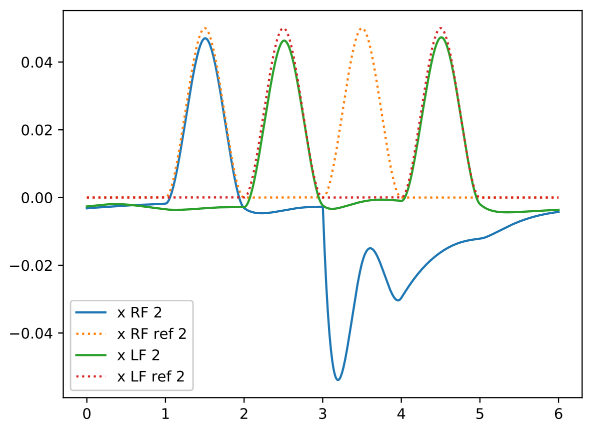


Figure 8: Squatted walk with vertical push, feet trajectory on the z axis; parameters: *w_com* $= 1 \cdot 10^3$, *w_foot* $= 2 \cdot 10^2$, *w_posture* $= 1 \cdot 10^{-1}$, *w_squat* $= 1 \cdot 10^5$, *kp_squat* $= 1 \cdot 10^1$, *kp_com* $= 1 \cdot 10^1$.

matically extract the generated figures and save them in *.eps* format to achieve higher quality in the report.

Lastly, we preferred to work with *meshcat* visualizer (instead of *Gepetto*) as we obtained better performance while running in *WSL2* (*Windows Subsystem for Linux*); still for the delivery we leave the default option of the visualizer.