University of Trento
Department of Industrial Engineering

Optimization Based Robot Control

Assignment 01

Humanoid walking with TSID

Professor:
Prof. Andrea Del Prete

Students:

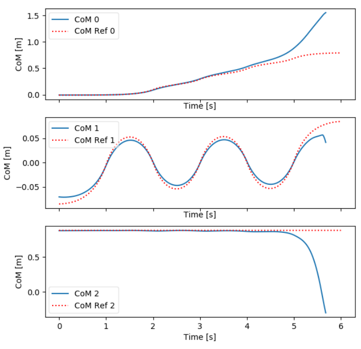
Lorenzo Colturato	lorenzo.colturato@studenti.unitn.it	233301	from DII
Luca Zardini	luca.zardini@studenti.unitn.it	229366	from DISI

Academic Year 2022-2023

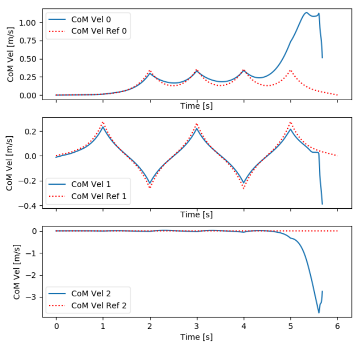
1st QUESTION

Simulations results. Default settings

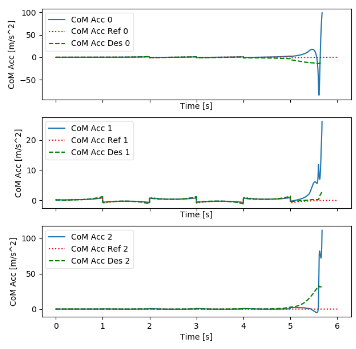
After the implementation of the function *"compute_ 3rd_ order_ poly_ traj()"*, explained in the code contained in the file *"hw1_ LIPM_ to_ TSID_ template.py"* and in *Appendix A* at the end of the report, the simulations on the walking humanoid were conducted, which can be seen in Figure 1, considering the default settings: SQUAT = 0, PUSH = 0, default weights and gains.



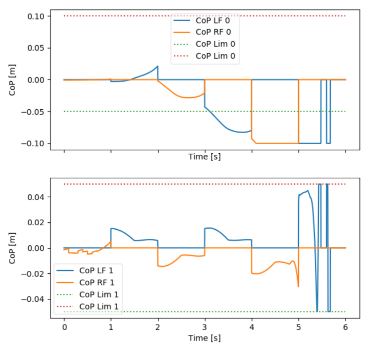
(a) CoM trajectory for x, y, z



(b) CoM velocity for x, y, z



(c) CoM acceleration for x, y, z



(d) CoP trajectory for RF and LF

Figure 1: Simulations results. Default settings

As can be seen from the Figures, at least up to 5 seconds of simulation, the results are quite good and consistent with the data of the reference motion of the humanoid. After 5 seconds (and after the robot has completed its path) the robot falls forward, resulting in a divergence in the trend of the position, speed and acceleration of its CoM with respect to the reference data. Furthermore, the x-trajectory of both feet exceeds the lower limit imposed at approximately -0.05 m. Another problem is that the robot's feet, from the second step onwards, begin to compenetrate with the ground below them.

Simulations results. Tuned weights

In tuning the weights we first tried to change each weight individually while keeping the others by default to see the specific effect of each weight on the robot's trajectory and what we found out is summarized in Table 1.

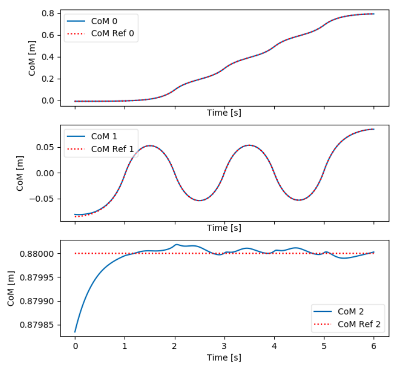
	Increase	**Decrease**
w_com	x,y traj. match the ref. traj. CoP x-limits transpassed over 1000 no more improvement	robot falls down very quickly no more able to do four steps 0.1 seems to be the limit
w_foot	feet do not compenetrate the ground better x/y traj., vel. and acc. over 100 robot falls down immediately	robot falls down easily no more able to do four steps
w_posture	robot falls down quickly no more able to do four steps CoP x-limits transpassed	x,y traj. very similar to ref. traj. CoP x-limits not transpassed if too low the robot loses equlibrium

Table 1: Effect of weights on robot's CoM trajectory, velocity and acceleration

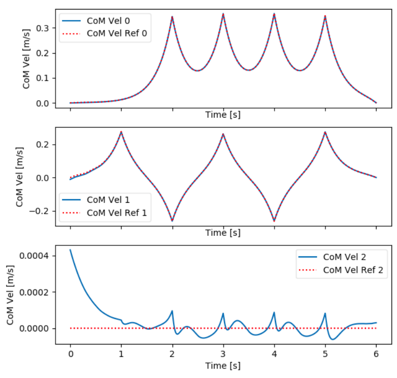
On the basis of these observations we performed tests by varying the weights and we finally arrived at a combination of values that allow to have better performance than the default values:

$$\begin{cases} \text{w\_com} = 5 \cdot 10^{-1} \\ \text{w\_foot} = 1 \cdot 10^{-1} \\ \text{w\_posture} = 2 \cdot 10^{-2} \end{cases} \xrightarrow{\text{from default to tuned}} \begin{cases} \text{w\_com} = 100 \\ \text{w\_foot} = 1 \\ \text{w\_posture} = 3 \cdot 10^{-4} \end{cases} \tag{1a}$$

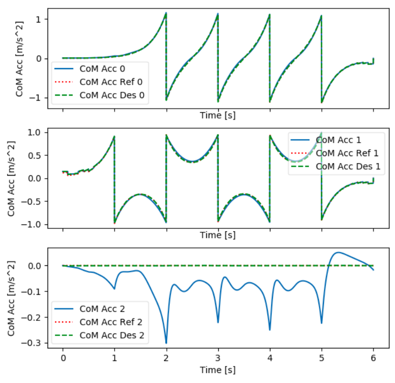
These weights result in a good motion of the humanoid, now able to perform the whole predefined path without falling, with consequent good simulation plots, shown in Figure 2. It can be seen that the robot's CoM x/y-trajectories, velocities and accelerations match the reference and the desired ones. Furthermore, the problem related to the compenetration of the feet with the ground is now solved, as well as the x-trajectory of the right and left foot that no longer exceed the lower limit imposed.



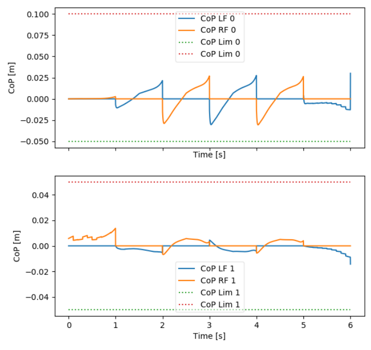
(a) CoM trajectory for x, y, z



(b) CoM velocity for x, y, z



(c) CoM acceleration for x, y, z



(d) CoP trajectory for RF and LF

Figure 2: Simulations results. Tuned weights

2nd QUESTION

Simulations results. SQUAT = 1, default weight and gain

Now we set SQUAT = 1 in the code to analyze the behavior of the robot, which performs a squat walk with our tuned weights and the default values of gain and weight for the squat task, that are w_squat = 10 and kp_squat = 10. Figure 3 shows the trajectory of the robot's CoM relative to the x-y-z axes.

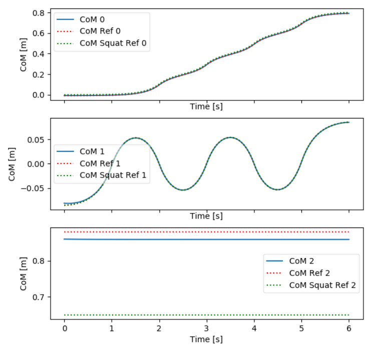
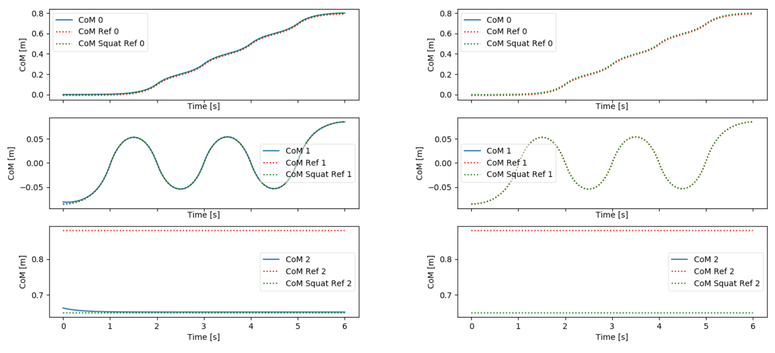


Figure 3: CoM trajectory for x, y, z. Default weight and gain

It can be noted that the z-trajectory remains constant around 0.85 m, a value quite distant from the reference value (green line) of 0.65 m. To bring the current trajectory closer to the reference trajectory of the squat walk, which means lowering the robot's CoM, it is necessary to increase the weight or gain parameters.

Simulations results. SQUAT = 1, increased weight and gain

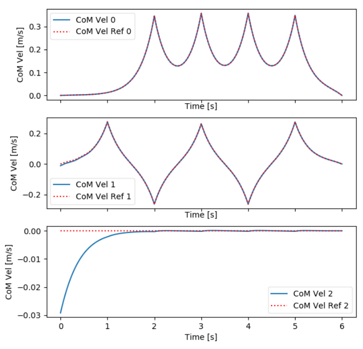
We tried to increase separately w_squat and kp_squat to see their effect on the robot's CoM height. First, we increased w_squat keeping kp_squat fixed. By doing this we noticed that, to reach the desired height of the CoM, it is necessary to increase w_squat up to approximately 10000, value that allows to keep the robot stable and balanced throughout the walk. On the contrary, we have seen that, when kp_squat=10000, the robot falls immediately during the stabilization phase, before lowering and walking. The simulations results on the z-trajectory of the robot's CoM are displayed in Figure 4.



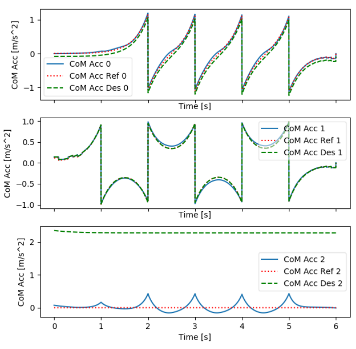
(a) w_squat = 10000 **(b)** kp_squat = 10000

Figure 4: CoM trajectory for x, y, z

The actual trajectory with kp_squat=10000 is not displayed, since the robot does not even starts walking, but falls down immediately. Figure 5 shows also the velocity and the acceleration of the robot's CoM for w_squat=10000.



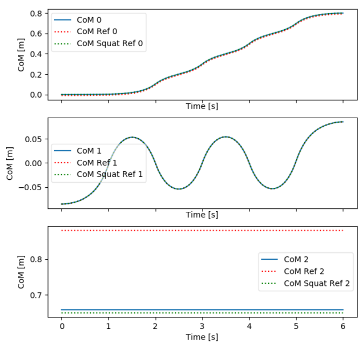
(a) CoM velocity for x, y, z



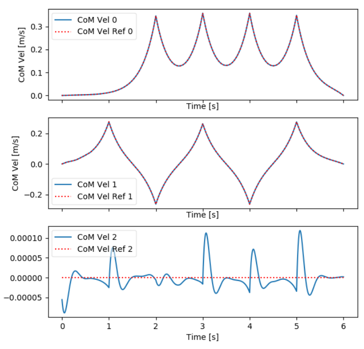
(b) CoM acceleration for x, y, z

Figure 5: w_squat=10000

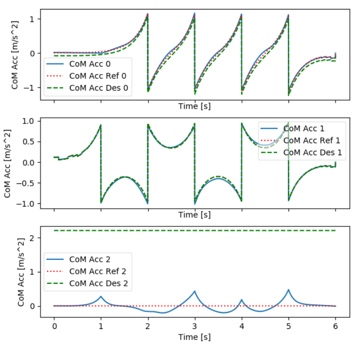
We have seen that the maximum value that we can assign to kp_squat is 2500. Higher values lead to the robot fall. However, with kp_squat=2500 the robot does not reach the reference height of the squat. Moreover, in lowering itself before walking, it suffers a loss of lateral balance. Figure 6 shows the robot's CoM movement for kp_squat = 2500.



(a) CoM trajectory for x, y, z



(b) CoM velocity for x, y, z



(c) CoM acceleration for x, y, z

Figure 6: Simulations results. kp_squat=2500

To eliminate the initial instability in lowering, it is necessary to use a gain equal to 100 for which the z-trajectory is approximately halfway between that desired in the squat walk and that desired for the normal walk, as can be seen in Figure 7.

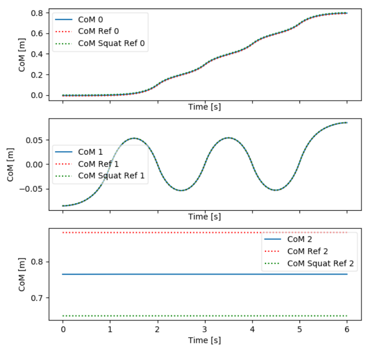


Figure 7: CoM trajectory for x, y, z. kp_squat=100

As can be seen there is a difference in changing weight and gain. In particular, the weight expresses how important the squat task is: the higher the weight, the more relevant the task. On the other hand, the gain is the proportional gain of the PD control law used by the controller. The PD control law is a feedback control system based on the error in the position and the velocity (proportional gain and derivative gain). If we increase k_p, the sensitivity of the controller to the error increases. As a consequence, it also increases the error compensation. Since k_p is a measure of the stiffness of the system, if it is too high the system becomes too stiff to the point that is unstable. Furthermore, this results in having a very responsive controller that tries to compensate for the error in a very short time: the robot lowers itself to the desired height of the CoM too quickly to the point that it falls in an attempt to reach the squat position, thus failing the task.

3rd QUESTION

Disabling the squat task and setting PUSH = 1 in the code, meaning that the robot will receive an intensity push of 0.1 m/s along the positive x-axis, and maintaining the same tuned weight of the 1^{st} question, the robot does not fall. Assuming it does, we thought the problem might lie in how the thrust of external action is handled by the controller. If the proportional gain of CoM task (kp_com) is too high (making the system unstable), the controller reacts to the thrust very quickly on a system which, however, is very rigid. If the thrust is such as to cause the robot to fall, the action of the controller, which rapidly tends to balance this force, is in any case not sufficient to prevent the fall of a body that is too rigid. As a result, we tried lowering kp_com, noting that the stability of the robot improves dramatically. Although the controller is less responsive, since the robot is less rigid, his action is such as to prevent it from falling and allow it to complete the walk. The disadvantage of this method lies precisely in the slower response of the controller to the position error that is generated by the effect of the thrust that the robot receives.

Another possible problem is that the default time used to stabilize the robot (T_pre), for high values of kp_com, may not be sufficient. It may be that, if a high gain (but low enough not to make the system too stiff) is used, with the default T_pre the robot will start walking even before it is fully balanced. If a thrust is applied in such conditions, since the robot is already unbalanced on its own, it will naturally falls. Therefore, increasing the settling time allows to use a higher gain. The advantage of using a higher gain is that the system becomes more responsive. The disadvantage of this solution is that, for real-time applications or applications that require actions close in time, this relatively long settling time negatively affects performance.

4th QUESTION

In this case the robot is subjected to a push of -0.5 m/s along negative z-axis, meaning that it is pushed down. Both the PUSH and SQUAT flags are enabled. Under these conditions the behavior of the robot is evaluated by considering two different combinations of values for w_squat and kp_squat. By simulating the walk with the two combinations of values and observing the trajectory along z, the axis that is most affected by the push, we have obtained some parameters, which are collected in Table 2, that allow to analyze the differences between the two different cases.

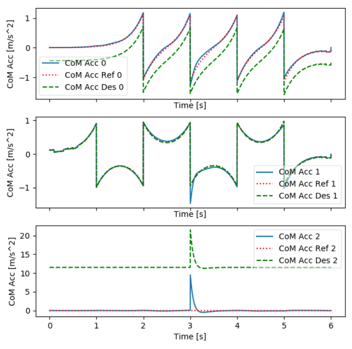
	w_squat = 100	**w_squat = 10**
	kp_squat = 100	**kp_squat = 1000**
Initial height [m]	0.76500	0.76500
Minimum height [m]	0.74684	0.75046
Time of minimum height [s]	3.1	3.078
Amount of shift [m]	0.0181	0.0145
Time to return to stability [s]	0.478	0.304
Max acceleration [m/s^2]	9.483	11.444

Table 2: Parameters related to the z-trajectory for different configurations of w_squat and kp_squat

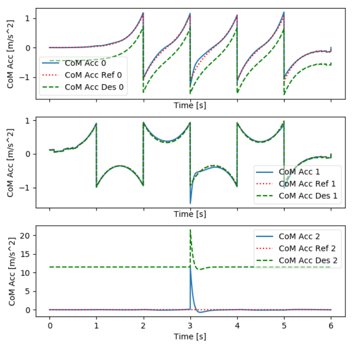
In particular, the amount of shift is the difference between the height of the CoM just before the push (at 3 s) and the minimum height of the trajectory. Instead, to get an approximation of the time it takes to return to stability, we decided to measure how many seconds it takes the robot, after the push, to bring its CoM back to near the height it had just before the push (we considered a deviation of $\pm$ 0.002 from the pre-push height).

As we expected, the system with a higher kp_squat is more responsive. This is evident by looking at the time of the minimum height: the robot receives a push at 3 s and, after 0.078 seconds, it is able to contrast this force. The Table shows also that the amount of time the system takes to return to the original position (*Time to return to stability*) is also lower. The maximum acceleration, which is related to the z direction, confirms that, by increasing kp_squat, the system acts faster to compensate for the generated error.

On the other hand, it is possible to see that a system with a lower gain is less responsive to the push and it takes 0.17 seconds more to return to the original z position. Figure 8 shows the difference in the CoM's acceleration along z axis between the two configurations of weight and gain.



(a) w_squat = 100, kp_squat = 100



(b) w_squat = 10, kp_squat = 1000

Figure 8: CoM acceleration for x, y, z

Appendix A

Computation of 3^{rd} order trajectory

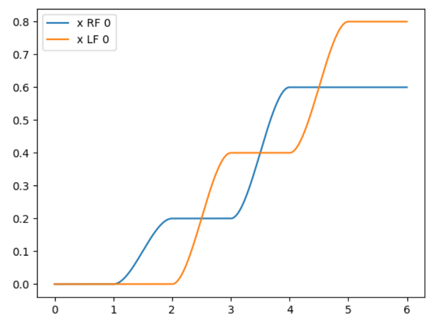
General third order equation:

$$x(t) = a + b \cdot t + c \cdot t^2 + d \cdot t^3$$

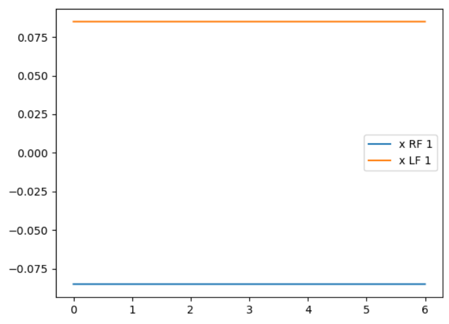
Imposing the following conditions: known initial position x_0, known final position x_1, null initial velocity and null final velocity, we came to a system of four equations in four unknowns of the type:

$$\begin{cases} x(t=0) = x_0 = a \\ x(t=T) = a + b \cdot T + c \cdot T^2 + d \cdot T^3 \\ \dot{x}(t=0) = b \\ \dot{x}(t=T) = b + 2c \cdot T + 3d \cdot T^2 \end{cases} \xrightarrow{\text{computed coefficients}} \begin{cases} a = x_0 \\ b = 0 \\ c = \frac{3(x_1 - x_0)}{T^2} \\ d = \frac{2(x_0 - x_1)}{T^3} \end{cases} \tag{A.1a}$$

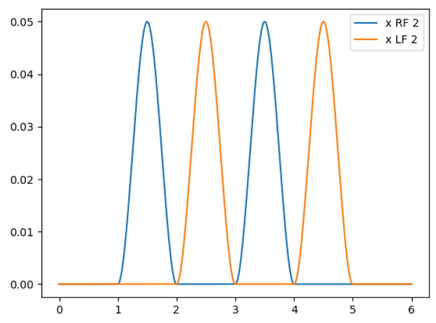
Figure A.1 shows the trajectories of the robot's feet along the three axes x, y and z computed with the implemented third order interpolating equation.



(a) x-trajectory



(b) y-trajectory



(c) z-trajectory

Figure A.1: Computed trajectories for the robot's feet