Assignment 01: Humanoid walking with TSID

Gianluigi Grandesso[*] - gianluigi.grandesso@unitn.it

October 2022

1 Description

The goals of this assignment are:

- making practice using Task Space Inverse Dynamics (TSID) to make a humanoid robot walk;
- understanding the role of the task weights and learning to tune them.

2 Submission procedure

You are encouraged to work on the assignments in groups of 2 people. **If you have a good reason to work alone, then you can do it, but this has to be previously validated by one of the instructors**. Groups of more than 2 people are not allowed. The mark of each assignment contributes to 10% of you final mark for the class (i.e. 3 points out of 30).

When you are done with the assignment, please submit a single compressed file (e.g., zip). **The file name should include the surnames of the group members**, and the file must contain:

- A pdf file with the answers to the questions, the **names and ID number** of the group members; you are encouraged to include plots and/or numerical values obtained through simulations to support your answers. **This pdf does not need to be long. One or two pages of text should be enough to answer the questions. You can then add other pages for plots and tables.**
- The complete orc folder containing all the python code that you have developed.

If you are working in a group (i.e., 2 people) only one of you has to submit.

Submitting the pdf file without the code is not allowed and would result in zero points. Your code should be consistent with your answers (i.e. it should be possible to produce the results that motivated your answers using the code that you submitted). If your code does not even run, then your mark will be zero, so make sure to submit a correct code.

[*]Optimization-based Robot Control, Industrial Engineering Department, University of Trento.

3 Tests on humanoid walking

We will use TSID to control a simulated humanoid (Talos by PAL Robotics) to make it walk in two different modes (normal walking and walking while squatting) and react to an external push. The code considers also a pre-phase of `T_pre` seconds needed to stabilize the humanoid before starting walking.

The reference foot and CoM trajectories needed to perform a 6-step walk are already computed (saved as `talos_walking_traj_lipm.npz`) using a Linear Inverted Pendulum Model (LIPM). However, the time step used in the reference trajectories is larger than the one of TSID, so you are asked, only for the foot trajectories, to implement a third order interpolating function to generate the new foot trajectories to be used as reference trajectories in TSID. This function, named `compute_3rd_order_poly_traj()`, must take the current LIPM foot position, next LIPM foot position, LIPM time step and TSID time step as inputs, and output the new sequence of position, velocity and acceleration values of the foot that considers the TSID time step. Thus, referring to the foot position trajectory as $x(t)$, you need to compute the coefficients a, b, c and d defining the third order polynomial:

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

The template code for the assignment is located in:
`code/orc/01_assignment/`

In this folder you will find:

- `tsid_biped.py` implementing the `TsidBiped` class that you have already practiced with during the lab session on 04/10
- `hw1_LIPM_to_TSID_template.py` containing the functions needed to transform the LIPM trajectories into reference trajectories discretized with a different timestep for TSID (where you need to implement `compute_3rd_order_poly_traj()`)
- `hw1_conf.py` defining all the parameters for both LIPM and TSID
- `hw1_tsid_biped_walking.py` that runs the simulation of the walking humanoid controlled with TSID

Once you have implemented the function `compute_3rd_order_poly_traj()`, you need to run `hw1_LIPM_to_TSID_template.py` just once before running `hw1_tsid_biped_walking.py` in order to generate and save the reference trajectories for TSID.

In the configuration file there are two flags, `SQUAT` and `PUSH`, that allows you to choose between a normal walk (`SQUAT`=0) and a squat walk (`SQUAT`=1) as well as applying a push (`PUSH`=1) to the humanoid CoM at half walk, whose direction and velocity are defined by the vector `push_robot_com_vel`. The squat version of the walk is implemented through an additional task that requires the robot to track a constant CoM height (`squat_height`) and the same trajectory used for the normal walk for the x and y CoM components.

Try to answer the following questions:

1. Run `hw1_tsid_biped_walking.py` with the default settings (`SQUAT`=0, `PUSH`=0 and default weights and gains). As you can notice, the humanoid falls down before completing the 6-steps walk. Tune the weights [`w_com`, `w_foot`, `w_posture`] to make the humanoid succeed in

performing the whole walk. Describe how you tuned the weights and what the effects of such change are. Please refer to the plots generated by the code to argue your answer.

2. Set `SQUAT`=1 using the weights that you just tuned. Comment the plot that shows the tracking of the CoM reference trajectory (focus on what happens to the tracking of the z component). If we want the robot to squat more and have its CoM closer to the desired height we could either increase `w_squat` or `kp_squat`. Could you tell the difference (if any) between the two approaches? If you try them, do you notice any difference in the plots? If yes, what causes those differences?

3. Set `SQUAT`=0 and `PUSH`=1 using your tuned weights. Does the robot fall down? If yes, what could you do to prevent it from doing it? Are there any drawbacks in doing what you suggested? If it does not fall, suppose it does and answer the previous questions.

4. Set `SQUAT`=0, `PUSH`=1 and `push_robot_com_vel`= $[0, 0, -0.5]$. Run two simulations, one with [`w_squat`=100, `kp_squat`=100] and another one with [`w_squat`=10, `kp_squat`=1000]. Comment on the differences that you notice and try to justify them.