LAB 6: Floating base robot: quasi-static control of locomotion stability

Notes:

- You can check the frequency at which the controller is publishing by inspecting the topic **/command** with **rostopic hz /command**.
- It is possible to set the verbosity level of the output, by setting the parameter *verbose* in `L6\_conf.py`.
- To have the experiment not ending after 5 s set the parameter $CONTINUOUS$=True in `L6\_conf.py` (can stop by hitting CTRL+C).

Important Note! if the frequency at which **/command** is not the right one (e.g. check the dt variable in the `base_controllers/params.py` for the robot that you selected) then it means your computer is not fast enough to run the controller, this can cause degraded performance with tracking errors that are not due to the implemented control strategy but to the delays! To mitigate these issues you can deactivate some parts that are superfluous and are demanding in terms of computational time:

- The most easy and effective option is to slow down the simulation calling the `p.startSimulator()` function
  with `world_name='slow.world'` argument, this slows down simulation frequency to 1/5th of the frequency (i.e. 200Hz). Note that the whole ROS environment will be slowed down with respect to the real world time, so also the controller, so with this trick it should be able to meet the required publishing frequency.
- Be sure you do not have any other program active (e.g. Zoom, Chrome)
- Comment the subscriber to the bumper plugins that computes the contact forces. Be sure the flag `p.use_ground_truth_contacts = False`
- do not publish any visual element (e.g. arrows) commenting the lines related to `p.ros_pub.add_arrow()` and `p.ros_pub.publishVisual()`

Exercises:
1) *Generate a Sinusoidal Reference for the Robot Trunk:* Generate a sinusoidal reference for Z direction ($f_z = 0.5\ Hz$, $A_z = 0.03\ rad$) together with another for the pitch direction ($f_\theta = 1\ Hz$, $A_\theta = 0.1\ rad$). Note that you will have to generate references also for velocity and acceleration. If you select a sin() you would have a non zero velocity at the beginning because the cos() that is its first derivative is non zero at t = 0. This could create tracking errors. Therefore is better to select a cos() for the position and shift it with an offset of -1 to make it pass for 0 at t = 0).

2) *Projection-based Quasi-Static Controller:* Design a controller for the *base frame* position and orientation with a projection-based approach (see utils/controlRoutines.py)).

2.1)Virtual Impedance: First design a virtual impedance to track the references generated in 1) for the base frame position and orientation. Compute the desired wrench $W^d = W_{fbk} \in \mathbb{R}^6$ to realize the virtual impedance:

$$W^d_{lin} = K_{lin}(x^d_b - x_b) + D_{lin}(\dot{x_b}^d - \dot{x}) \tag{1}$$

$$W^d_{ang} = -K_{ang}e_o + D_{ang}(\dot{\omega_b}^d - \omega) \tag{2}$$

where $e_o \in \mathbb{R}^3$ is the orientation error.

2.2) Mapping to torques: Map the desired wrench into torques (with quasi-static assumption):

$$\tau^d = -J^T_{cj}(J^T_b)^\dagger W^d \tag{3}$$

where:

$$J^T_b = \begin{bmatrix} I_{3\times3} & \dots & I_{3\times3} \\ [x_{f_1} - x_b]_\times & \dots & [x_{f_c} - x_b]_\times \end{bmatrix} \tag{4}$$

Note that $W = J^T_b f$ represents the contact wrench that is given as input to the Newton-Euler equations, but we are doing an approximation here: we are controlling the orientation of the base link rather than an equivalent rigid body whose angular velocity $\bar{\omega}$ is the *average* angular velocity of all the links of the robot (as Newton-Euler equations state). In essence we are approximating $\bar{\omega}$ with ω_b. Check the robot is tracking the sinusoidal reference trajectory of point 1). See that without compensating gravity the robot is barely able to stand-up accumulating huge errors on the Z direction.
3) *Gravity compensation:* Add a gravity term W_g to the desired wrench W^d.

$$W^d = W_{fbk} + W_g \tag{5}$$

How does W_g look like? (hint: remember that gravity is applied to CoM). Which changes are necessary if we close the loop at the base frame?

$$W_g = \begin{bmatrix} mg \\ x_{b,com} \times mg \end{bmatrix} \tag{6}$$

How is the tracking error in steady state? and in motion? Play with the frequency of the controller (e.g. check parameter dt) and see how the system can tolerate (without being unstable) higher damping values for higher loop frequencies.

4) *Feed-forward term:* Add a feed-forward term the desired wrench W^d.

$$W^d = W_{fbk} + W_g + W_{ffwd} \tag{7}$$

How would you implement that at the impedance level? (hint: exploit the mass matrix and the tensor of inertia). Be sure that at the angular level you use $\dot{\omega}^d$, and not $\ddot{\Phi}^d$. How is the tracking of the CoM and of the trunk orientation improved during motion? Compare also the tracking of ground reaction forces with and without feed-forward term.

5) *Control of CoM:* Now we want to close the loop at the CoM, because it is the relevant point for locomotion stability. Modify the controller to control the position of the CoM in place of the base frame origin. Which changes are necessary?

$$J_{com}^T = \begin{bmatrix} I_{3\times3} & \dots & I_{3\times3} \\ [x_{f_1} - x_{b,com}]_\times & \dots & [x_{f_c} - x_{com}]_\times \end{bmatrix} \tag{8}$$

Note that now the gravity compensation simplifies to $W_g = \begin{bmatrix} mg \\ 0_{3\times1} \end{bmatrix}$ Add 5 cm uncertainty (this is the shift due to the presence of an arm attached to the front of the robot) to the CoM offset $x_{b,com}$ in the X direction and see how the tracking deteriorates.

6) *Check static stability:* Now comment the sinusoidal reference generation and design an horizontal trajectory for the CoM (e.g. moving left along Y direction) starting from the default configuration q_0. What happens when the CoM goes out of the polygon? Plot the ground reaction forces and check they are going to zero in the RF and RH leg and the whole weight is only on two legs.

7) *Quasi-static QP controller:* Re-implement the previous mapping from wrench to contact forces $f^d \in \mathbb{R}^k$ by casting it as an optimization problem (QP) that optimizes for these forces. Enforce *unilateral* constraints for the legs that are in stance $f_{z,min} = 0$.

$$\begin{aligned} f^d = &\underset{f\in\mathbb{R}^k}{\operatorname{argmin}} \|W - W^d\| \\ &s.t. \quad \underline{d} < Cf < \bar{d}, \end{aligned} \tag{9}$$

where in the cost term we are trying to track the desired wrench W^d. This can be expanded into:

$$\begin{aligned} f^d = &\underset{f\in\mathbb{R}^k}{\operatorname{argmin}} (Af - b)^T S(Af - b) \\ &s.t. \quad \underline{d} < Cf < \bar{d}, \end{aligned} \tag{10}$$

where:

$$C = \begin{bmatrix} C_0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & C_c \end{bmatrix} \quad \underline{d} = \begin{bmatrix} \underline{d}_0 \\ \vdots \\ \underline{d}_c \end{bmatrix} \quad \bar{d} = \begin{bmatrix} \bar{d}_0 \\ \vdots \\ \bar{d}_c \end{bmatrix}, \tag{11}$$

with: $A = J_b^T$ and $b = W^d$, $C_i = n_i^T$, $\underline{d}_i = f_{min_i}$, $\bar{d}_i = f_{max_i}$. The $S \in \mathbb{R}^6$ is a selection matrix that allows you to promote the tracking in certain directions rather that others when the constraints are hit.

Is there something missing in this optimization? There is not unique solution and this is due to the fact it exists a null-space for A, that makes the cost positive semi-definite. What is its dimension? Try to add a regularization term with adjustable gain α to make the Hessian of the cost positive definite:

$$\begin{aligned} f^d = & \underset{f \in \mathbb{R}^k}{\operatorname{argmin}} (Af - b)^T S (Af - b) + \alpha f^T W f \\ & s.t. \quad \underline{d} < Cf < \bar{d}, \end{aligned} \tag{12}$$

a) Compare with the previous projection-based controller of point 6). b) Does it tolerates higher frequencies on the reference trajectories? c) Check the Z component of the ground reaction forces is always positive. Does it makes sense to set f_{min} exactly at zero? or it is safer to allow a bit of positive margin? why? Try to play with the f_{min} increasing it for just one leg and verity that the constraint on the Z component is not violated for that leg.

8) *Some experiences with the quasi-static QP controller:*

a) Try to set a static target for the CoM inside the triangle formed by the LF, RF and LH leg (need to comment sinusoidal reference). b) After 1.5 s remove the RH leg from the set of stance legs. How the load is redistributed on the other three legs? Check the contact force goes to zero for that leg and the robot stays in equilibrium on only 3 legs.

9) *Friction cones:* Did you notice some slippage at the feet while following the sinusoidal reference? Modify the C_i matrix to add the friction cone constraints for each stance leg, with friction coefficient μ_i:

$$C_i = \begin{bmatrix} (t_{x_i} - \mu_i n_i)^T \\ (-t_{x_i} - \mu_i n_i)^T \\ (t_{y_i} - \mu_i n_i)^T \\ (-t_{y_i} - \mu_i n_i)^T \end{bmatrix} \quad \bar{d}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{13}$$

Note that the unilaterality constraints are naturally incorporated in the friction cones and are no longer needed. Run again a simulation setting a conservative friction coefficient of 0.7 (the Gazebo simulator use 0.8 by default). a) Is the slippage still happening? has the tracking error increased? b)Plot the constraint violations to check is there is any. Try

to reduce the friction coefficient and observe the number constraint violations increases. c) Try to tilt the cones setting the normals at the feet pointing 45 degs inward (not implemented in the code). See the robustness becomes low because the contact forces are lying on the boundary of the cone. d) reduce the friction coefficient and plot the constraint violations seeing they are changing.

10) *Friction cones and robustness (not implemented in the code):*

Modify the regularization term. Exploit the force redundancy to regularize the contact forces to stay in the middle of the cones. Set the regularization matrix to:

$$W = \begin{bmatrix} {}_wR_1 W_{n_0} {}_wR_1^T & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & {}_wR_c W_{n_c} {}_wR_c^T \end{bmatrix} \tag{14}$$

where ${}_wR_i = \begin{bmatrix} t_{x_i} & t_{y_i} & n_i \end{bmatrix}$ is the rotation matrix that maps vector from the contact frame of leg i to the world frame. Set $W_{n,xy}$ higher to penalize tangential forces in the contact frame. What is the size of the internal force redundancy with 4 legs on the ground? Now modify the regularization term to minimize joint torques (e.g. $\tau^T\tau$):

$$W = J_{cq} W_\tau J_{cq}^T \tag{15}$$

with $W_\tau \in \mathbb{R}^n$ is the weighting matrix for the torque vector.