Récapitulatif de la méthode MPC 09/04/2020

See `https://arxiv.org/pdf/1909.06586.pdf` for the original MIT article "Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control"

1 State Estimator

The role of the state estimator is to provide an estimation of the position, orientation, linear velocity and angular velocity of the base of the quadruped as well as the angular position of the actuators. In the case of a simulation with PyBullet this information is perfectly known and can be retrieved with the API.

The position/orientation state vector of the quadruped in world frame is:

$$ {}^{o}q = \begin{bmatrix} {}^{o}x & {}^{o}y & {}^{o}z & {}^{o}a & {}^{o}b & {}^{o}c & {}^{o}d & \theta_0 & \dots & \theta_{11} \end{bmatrix}^T \tag{1} $$

With $({}^{o}a, {}^{o}b, {}^{o}c, {}^{o}d)$ the quaternion associated with the orientation of the base in world frame (usually written (x, y, z, w) but x, y and z are already used for the position). θ_0 to θ_{11} are the angular positions of the 12 actuators of the quadruped.

The velocity state vector of the quadruped in world frame is:

$$ {}^{o}\dot{q} = \begin{bmatrix} {}^{o}\dot{x} & {}^{o}\dot{y} & {}^{o}\dot{z} & {}^{o}\omega_x & {}^{o}\omega_y & {}^{o}\omega_z & \dot{\theta}_0 & \dots & \dot{\theta}_{11} \end{bmatrix}^T \tag{2} $$

With $({}^{o}\omega_x, {}^{o}\omega_y, {}^{o}\omega_z)$ the angular velocities about the x, y and z axes of the world frame.

2 MpcInterface

The role of the MpcInterface object is to transform data coming from PyBullet (simulation) or the state estimator of the robot (real world) into useful information for the rest of the control loop.

Data coming from PyBullet is retrieved in the world frame o. The position of the base of the quadruped in this frame can be noted $[{}^{o}x\ {}^{o}y\ {}^{o}z]^T$ and its orientation $[{}^{o}\phi\ {}^{o}\theta\ {}^{o}\psi]^T$ with (ϕ, θ, ψ) the roll, pitch and yaw angles (Tait-Bryan Euler angles). These angles corresponds to a sequence of rotations about the z, then y and then x axis such that the transform from body to world coordinates can be expressed as:

$$ R = R_z(\psi) R_y(\theta) R_x(\phi) \tag{3} $$

Position, orientation, linear velocity and angular velocity of the base of the quadruped in world frame can be transformed either into the base frame b or into the local frame l, as defined in Figure X. The transform between two frames 1 and 2 can be stored in an object ${}^{1}M_2$ that

contains the translation part ${}^1T_2$ and the rotation part ${}^1R_2$ of the transform. The relation between position $[{}^2x\ {}^2y\ {}^2z]^T$ in frame 2 and the same position in frame 1 is:

$$\begin{bmatrix} {}^1x \\ {}^1y \\ {}^1z \end{bmatrix} = {}^1R_2 \cdot \begin{bmatrix} {}^2x \\ {}^2y \\ {}^2z \end{bmatrix} + {}^1T_2 = {}^1M_2 \cdot \begin{bmatrix} {}^2x \\ {}^2y \\ {}^2z \end{bmatrix} \tag{4}$$

Based on Figure X the transforms are defined as follows:

$$\begin{aligned} {}^oT_b &= \begin{bmatrix} {}^ox & {}^oy & {}^oz \end{bmatrix}^T && (5)\\ {}^oR_b &= R_3({}^o\phi) \cdot R_3({}^o\theta) \cdot R_3({}^o\psi) && (6)\\ {}^oT_l &= \begin{bmatrix} {}^ox & {}^oy & {}^oz_{min} \end{bmatrix}^T && (7)\\ {}^oR_l &= R_3({}^o\psi) && (8) \end{aligned}$$

${}^oz_{min}$ is the altitude of the lowest feet in world frame i.e. the z coordinate of its center in this frame. Position of feet in world frame are retrieved from PyBullet.

To get the position and velocity of the center of mass (CoM) of the quadruped, Pinocchio requires the position and orientation of the base in world frame, the angular positions of the actuators and the linear and angular velocities of the base in base frame. All of them are directly retrieved from PyBullet except the linear and angular velocities ${}^bV_{base}$ and ${}^bW_{base}$ in base frame:

$$\begin{aligned} {}^bV_{base} &= ({}^oR_b)^{-1} \cdot {}^oV_{base} && (9)\\ {}^bW_{base} &= ({}^oR_b)^{-1} \cdot {}^oW_{base} && (10) \end{aligned}$$

The resulting position and linear velocity of the CoM in world frame are noted ${}^oC$ and ${}^oV$. The angular velocity in world frame ${}^oW$ is directly retrieved from PyBullet (assumption that ${}^oW = {}^oW_{base}$), just like the orientation ${}^oRPY = [{}^o\phi\ {}^o\theta\ {}^o\psi]^T$.

The position, orientation, linear velocity and angular velocity of the base of the quadruped in local frame can be retrieved using the transform ${}^oM_l$:

$$\begin{aligned} {}^lC &= ({}^oM_l)^{-1} \cdot {}^oC && (11)\\ {}^lRPY &= [{}^o\phi\ \ {}^o\theta\ \ 0]^T && (12)\\ {}^lV &= ({}^oR_l)^{-1} \cdot {}^oV && (13)\\ {}^lW &= ({}^oR_l)^{-1} \cdot {}^oW && (14) \end{aligned}$$

The projections on the ground of the shoulders of the quadruped are supposed constant even if in practice there is a dependence to roll and pitch. Order of shoulders is Front-Left, Front-Right, Hind-Left, Hind-Right:

$$\begin{aligned} {}^lshoulders &= \begin{bmatrix} 0.19 & 0.19 & -0.19 & -0.19 \\ 0.15005 & -0.15005 & 0.15005 & -0.15005 \\ 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix} && (15)\\ {}^oshoulders &= {}^oM_l \cdot {}^lshoulders && (16) \end{aligned}$$

Positions of feet in world frame ${}^or$ are directly retrieved from PyBullet and transformed in local frame for the MPC:

$$ {}^lr = ({}^oM_l)^{-1} \cdot {}^or \tag{17}$$

3 Footstep Planner

The desired gait for the quadruped is defined in a gait matrix of size 6 by 5. Each row contains information about one phase of the gait. The first column contains the number of remaining time steps of the MPC for each phase and the four remaining columns contains the desired contact status for each foot and for each phase (0 if the foot is in swing phase or 1 if it is in stance phase).

With a time step of 0.02 s for the MPC and a gait period of 0.32 s, the matrix of a walking trot gait is defined as follows:

$$gait(0) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 7 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 7 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{18}$$

The first phase is a 4-feet stance phase that lasts 1 iteration of the MPC, followed by a phase with 2 feet in stance phase and the other 2 in swing phase during 7 iterations, then again a 4-feet stance phase and finally 2 feet in stance phase and 2 feet in swing phase. As the quadruped moves forward in the gait, the gait matrix undergoes a rolling process. For instance after 3 iterations of the MPC this matrix becomes:

$$gait(1) = \begin{bmatrix} 7 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 7 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad gait(2) = \begin{bmatrix} 6 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 7 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad gait(3) = \begin{bmatrix} 5 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 7 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{19}$$

Additional rows could be added to be able to handle more complex gaits.

The gait being defined, let's describe the way the location of future footsteps is computed. The footstep planner only works in 2 dimensions as it outputs the desired position $(r^{x,\star}, r^{y,\star})$ of each foot on the ground which is assumed flat. It only considers as input an horizontal linear reference velocity with an angular reference velocity about the vertical axis: $(\dot{x}^\star, \dot{y}^\star, \omega_z^\star)$.

The default position of footsteps in local frame is on the ground under the shoulders:

$$r_{sh} = {}^{l}shoulders = \begin{bmatrix} 0.19 & 0.19 & -0.19 & -0.19 \\ 0.15005 & -0.15005 & 0.15005 & -0.15005 \end{bmatrix} \tag{20}$$

A symmetry term is added to this position to make the gait more symmetric compared to the shoulders when moving. If the base moves forwards at speed v then if a foot lands under its associated shoulder it will spend the whole stance phase "behind" the shoulder as the base keeps moving forwards while the foot in contact does not move (in world frame). During the duration of the stance phase, the displacement of the base is equal to $t_{stance}v$. That is why with the symmetry term trying to land $\frac{t_{stance}}{2}v$ in front of the shoulder feet in contact spend half the stance phase in front of the shoulder and the other half behind it.

$$r_{sym} = \frac{t_{stance}}{2} \, {}^{l}v = \frac{t_{stance}}{2} \begin{bmatrix} {}^{l}\dot{x} \\ {}^{l}\dot{y} \end{bmatrix} \tag{21}$$

A feedback term is added to the footstep planner to make it easier for the robot to reach the reference velocity. The only way the quadruped can interact with its environment is by pushing on the ground with its feet (cannot pull). As per Newton's second law, if the quadruped wants to move in a given direction it has to apply a force in the inverse direction. So the feedback term makes it easier to do that by shifting the desired location of footsteps in the inverse direction of the velocity error $(v^\star - v)$. For instance if the robot is not moving fast enough forwards then the feedback term will slightly shift the footsteps backwards so that it's easier to push on the ground backwards and as a result to increase its forward velocity.

$$r_{fb} = k\ ({}^{l}v - {}^{l}v^\star) = k \begin{bmatrix} {}^{l}\dot{x} - {}^{l}\dot{x}^\star \\ {}^{l}\dot{y} - {}^{l}\dot{y}^\star \end{bmatrix} \tag{22}$$

The feedback gain k is equal to 0.03 (MIT's value).

A centrifugal term is added to the footstep planner to make it easier to compensate the centrifugal effect when the robot is turning about the vertical axis by adjusting the location of footsteps accordingly. As the formula involves the desired angular speed rather than the current one, it could also be seen as a way to help the quadruped reach the reference angular velocity in a way similar to what the feedback term does for the linear velocity.

$$r_c = \frac{1}{2}\sqrt{\frac{h}{g}}\ {}^{l}v \times {}^{l}\omega^\star = \frac{1}{2}\sqrt{\frac{h}{g}} \begin{bmatrix} {}^{l}\dot{y}\ \omega^\star_{l_z} \\ -{}^{l}\dot{x}\ \omega^\star_{l_z} \end{bmatrix} \tag{23}$$

Finally, another term is added to the footstep planning to take into account a temporal aspect. With all previous terms, there is none: whether a foot is just at the start of its swing phase or almost at the end, the desired target location returned by the footstep planner is the same. If the quadruped is moving forwards at the reference velocity then during the whole duration of a swing phase the target position will be Δx meters in front of the shoulder in local frame. Except since the base is moving in world frame then the target position in world frame is moving as well. At the start of each swing phase the associated foot will target a position $x_0 + \Delta x$ in world frame but by the end of the swing phase this position becomes $x_0 + \Delta x + t_{swing}\ \dot{x}^\star$ due to the movement of the base. With the assumption that current and reference velocities do not change much over one period of gait feet could directly aim for their final target location by taking into account the movement of the base during their swing phase.

With the assumption that the quadruped moves with constant linear and angular velocities during the remaining duration of the swing phase then the predicted movement is, if $\omega_{l_z} \neq 0$:

$${}^{l}x_{pred}(t_r) = \int_0^{t_r} \left({}^{l}\dot{x} \cos(\omega_{l_z}\ t) - {}^{l}\dot{y} \sin(\omega_{l_z}\ t) \right)\ dt \tag{24}$$

$${}^{l}y_{pred}(t_r) = \int_0^{t_r} \left({}^{l}\dot{x} \sin(\omega_{l_z}\ t) + {}^{l}\dot{y} \cos(\omega_{l_z}\ t) \right)\ dt \tag{25}$$

$${}^{l}x_{pred}(t_r) = \frac{{}^{l}\dot{x} \sin(\omega_{l_z}\ t_r) + {}^{l}\dot{y}\ (\cos(\omega_{l_z}\ t_r) - 1)}{\omega_{l_z}} \tag{26}$$

$${}^{l}y_{pred}(t_r) = \frac{-{}^{l}\dot{x}\ (\cos(\omega_{l_z}\ t_r) - 1) + {}^{l}\dot{y} \sin(\omega_{l_z}\ t_r)}{\omega_{l_z}} \tag{27}$$

If $\omega_{l_z} = 0$:

$${}^{l}x_{pred}(t_r) = {}^{l}\dot{x}\ t_r \tag{28}$$

$${}^{l}y_{pred}(t_r) = {}^{l}\dot{y}\ t_r \tag{29}$$

The remaining duration t_r for the swing phase of a foot can be directly retrieved using information contained in the gait matrix like the contact status (0 for a swing phase) and the remaining number of time steps (first column).

The desired location of footsteps is the sum of all described terms. Symmetry, feedback and centrifugal terms are the same for all feet contrary to the shoulder and prediction terms.

The desired location of footsteps for each gait phase in the prediction horizon are stored in a 6 by 13 matrix (same number of rows than the gait matrix). The first column is the same and contains the remaining number of footsteps for each phase. The twelve other columns contains the desired location of the footstep (if in stance phase) or Not-A-Number (if in swing phase) for the 4 feet. For instance for the $gait(1)$ matrix of equation 19 this matrix would be:

$$\begin{bmatrix} 7 & {}^{l}r_0^x & {}^{l}r_0^y & {}^{l}r_0^z & NaN & NaN & NaN & NaN & NaN & NaN & {}^{l}r_3^x & {}^{l}r_3^y & {}^{l}r_3^z \\ 1 & {}^{l}r_0^x & {}^{l}r_0^y & {}^{l}r_0^z & {}^{l}r_1^{x,\star} & {}^{l}r_1^{y,\star} & 0 & {}^{l}r_2^{x,\star} & {}^{l}r_2^{y,\star} & 0 & {}^{l}r_3^x & {}^{l}r_3^y & {}^{l}r_3^z \\ 7 & NaN & NaN & NaN & {}^{l}r_1^{x,\star} & {}^{l}r_1^{y,\star} & 0 & {}^{l}r_2^{x,\star} & {}^{l}r_2^{y,\star} & 0 & NaN & NaN & NaN \\ 1 & {}^{l}r_0^{x,\star} & {}^{l}r_0^{y,\star} & 0 & {}^{l}r_1^{x,\star} & {}^{l}r_1^{y,\star} & 0 & {}^{l}r_2^{x,\star} & {}^{l}r_2^{y,\star} & 0 & {}^{l}r_3^{x,\star} & {}^{l}r_3^{y,\star} & 0 \\ 0 & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN \\ 0 & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN & NaN \end{bmatrix}$$

The first row is a phase with two feet in swing phase (feet 1 and 2) and two feet in stance phase (feet 0 and 3). Feet 1 and 2 in swing phase receive a NaN value while feet 0 and 3 in stance phase receive their desired location. Since the first row is the current active phase the desired position of feet in stance phase is their current position in local frame. Due to how ${}^{o}M_l$ has been defined in equation 7 and 8 and the fact that ${}^{l}r = ({}^{o}M_l)^{-1} \cdot {}^{o}r$ that means ${}^{l}r_0^z = {}^{o}r_0^z - {}^{o}z_{min}$. For a ground that is completely flat ${}^{o}r_0^z = {}^{o}r_3^z = {}^{o}z_{min}$ so ${}^{l}r_0^z = {}^{l}r_3^z = 0$. However if foot 0 is on a small step then ${}^{o}r_3^z = {}^{o}z_{min}$ and ${}^{o}r_0^z = {}^{o}z_{min} + h_{step}$ so ${}^{l}r_3^z = 0$ and ${}^{l}r_0^z = h_{step}$. That way the fact that foot 0 is not strictly on the ground but on a small step is taken into account.

For the second row (the next phase), feet 0 and 3 are still in stance phase so their desired position is still their current position. For feet 1 and 2 that will be in stance phase (they are currently in swing phase) their desired position is the one outputted by the footstep planner. It only provides desired position for the x and y components so for z the assumption is made that the ground is flat (${}^{l}r_1^{z,\star} = {}^{l}r_2^{z,\star} = 0$) since there is no *a priori* knowledge about the environment.

For the third phase, feet 0 and 3 are now in swing phase so they have a NaN value while feet 1 and 2 are still in stance phase compared to the previous phase so their desired positions do not change. For the fourth phase feet 0 and 3 are back in stance phase. Their current position is not used since this stance phase happens in the future after a swing phase so instead their desired position is the one outputted by the footstep planner.

4 Foot trajectory generator

x, y and z in this section replace ${}^{o}r^x$, ${}^{o}r^y$ and ${}^{o}r^z$ for clarity purpose.

During swing phases feet have to be guided from their current position to their target position on the ground outputted by the footstep planner. To generate their trajectory in the air, foot trajectory generators are used, one for each foot. Each generator takes as input the current state of its associated foot $[x_{ft}\ \dot{x}_{ft}\ \ddot{x}_{ft}\ y_{ft}\ \dot{y}_{ft}\ \ddot{y}_{ft}]$, the desired position on the ground $[x_{goal}\ y_{goal}]$, the control time step dt, the time t_0 elapsed since the start of the swing phase and the expected duration t_1 of the swing phase. This information is processed to output a command

$[x^\star\ \dot{x}^\star\ \ddot{x}^\star\ y^\star\ \dot{y}^\star\ \ddot{y}^\star\ z^\star\ \dot{z}^\star\ \ddot{z}^\star]$ for the foot. Generators work in the world frame and use data provided by the MpcInterface.

For the x component, what the generator does is to tune a 5-th polynomial to have $x(t_0) = x_{ft}$, $\dot{x}(t_0) = \dot{x}_{ft}$ and $\ddot{x}(t_0) = \ddot{x}_{ft}$ while having $x(t_1) = x_{goal}$, $\dot{x}(t_1) = \ddot{x}(t_1) = 0$. The generator can then output $[x^\star\ \dot{x}^\star\ \ddot{x}^\star]$ by computing $x(t_0 + dt)$, $\dot{x}(t_0 + dt)$ and $\ddot{x}(t_0 + dt)$. The same happens for the y component to output $[y^\star\ \dot{y}^\star\ \ddot{y}^\star]$.

Command for the z component is deterministic and there is no feedback like for the x and y components for which the current position, velocity and acceleration of the foot are taken into account. Trajectory for z is a 6-th order polynomial that does not change and defined in such a way that $z(0) = \dot{z}(0) = \ddot{z}(0) = 0$ and $z(t_1) = \dot{z}(t_1) = \ddot{z}(t_1) = 0$ with $z(\frac{t_1}{2}) = h$. h is a constant value defined when the foot trajectory generator is created to set the desired apex height of the foot during its swing phase.

Due to these characteristics, the trajectory generated can be described as a bell-like trajectory that goes from the initial position of the foot to its target trajectory while respecting non-slipping constraints during take-off and landing (no horizontal speed) and trying to land softly (0 final velocity and acceleration for z).

To keep this slipping-avoidance property, the target position on the ground $[x_{goal}\ y_{goal}]$ is locked t_{lock} seconds before landing. Basically $[x_{goal}\ y_{goal}]$ is not updated if $t_0 > (t_1 - t_{lock})$. Changing the desired position on the ground just before landing would create a non-negligible horizontal speed to correct the position of the foot in order to land at the new position. It is required because the target position is always changing since it is linked to the current velocity of the robot through the symmetry term of the footstep planner and this velocity is never exactly the same from one time step to another.

Since a 3D tracking task is in charge of applying the adequate torques to follow the command of the trajectory generator depending on the current state of the foot (more details in the Inverse Dynamics section), there is already a feedback for the position, velocity and acceleration of the foot. To avoid having two feedback loops that try to do the same thing, the feedback of the foot trajectory generator is not used. What that means is that at the start of the swing phase the trajectory generator receives $[x_{ft}\ 0\ 0\ y_{ft}\ 0\ 0]$ to update the position of the foot and then the command of the generator is supposed to be perfectly followed. Therefore at the next iteration the generator is given the command $[x^\star\ \dot{x}^\star\ \ddot{x}^\star\ y^\star\ \dot{y}^\star\ \ddot{y}^\star]$ as the state of the foot. That way the feedback capabilities of the generator are not used.

As explained earlier, the trajectory for the z component is deterministic: it always starts at $z(0) = 0$ m and ends at $z(t_1) = 0$ m. That is why a small offset is added to the command $z^\star$ that is sent to the 3D tracking task to take into account the altitude of the ground the robot is walking on. This z_{offset} is determined by taking the minimum altitude of all feet in contact with the ground which uses the assumption that there is at least one feet in contact when the other feet are in swing phase (equal to ${}^{o}z_{min}$ introduced in the MpcInterface section). With this offset the command received by the tracking task will start and end at the correct altitude.

5 State vector

The reference velocity $\dot{q}^\star$ that is sent to the robot is expressed in its local frame. It has 6 dimensions: 3 for the linear velocity and 3 for the angular one.

$$ {}^{l}\dot{q}^\star = [{}^{l}\dot{x}^\star \;\; {}^{l}\dot{y}^\star \;\; {}^{l}\dot{z}^\star \;\; \omega_{l_x}^\star \;\; \omega_{l_y}^\star \;\; \omega_{l_z}^\star]^T \tag{30} $$

The velocity vector of the robot is:

$$ {}^{l}\dot{q} = [{}^{l}\dot{x} \;\; {}^{l}\dot{y} \;\; {}^{l}\dot{z} \;\; \omega_{l_x} \;\; \omega_{l_y} \;\; \omega_{l_z}]^T \tag{31} $$

At the start each iteration of the MPC, the current position and orientation of the robot defines a new frame in which the solver will work. This frame is a copy of the local frame so it is at ground level with the x axis pointing forwards (x axis of the local frame), the y axis pointing to the left (y axis of the local frame) and the z axis point upwards. Instead of working in terms of rotation around the x, y and z axes of the world frame, the solver will work with the pitch, roll and yaw angles defined in this new frame. The solver is working in a copy of the local frame, initial conditions of the solving process are as follows:

$$ q_0 = [{}^{l}x \;\; {}^{l}y \;\; {}^{l}z \;\; {}^{l}\phi \;\; {}^{l}\theta \;\; {}^{l}\psi]^T = [{}^{l}C_x \;\; {}^{l}C_y \;\; {}^{l}C_z \;\; {}^{l}\phi \;\; {}^{l}\theta \;\; 0]^T \tag{32} $$

$$ \dot{q}_0 = [{}^{l}\dot{x} \;\; {}^{l}\dot{y} \;\; {}^{l}\dot{z} \;\; \omega_{l_x} \;\; \omega_{l_y} \;\; \omega_{l_z}]^T \tag{33} $$

With $[{}^{l}C_x \;\; {}^{l}C_y \;\; {}^{l}C_z]$ the position of the center of mass in local frame.

The state vector of the robot and the reference state vector are then:

$$ X = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \qquad X^\star = \begin{bmatrix} q^\star \\ \dot{q}^\star \end{bmatrix} \tag{34} $$

The reference velocity is supposed constant over the prediction horizon in the local frame of the robot so it has to be properly rotated to be consistent with its future orientation.

For time step k of the prediction horizon, the reference velocity vector is defined as follows:

$$ \forall k \in [1, n_{steps}], \;\; \dot{q}_k^\star = \begin{bmatrix} R_z(\Delta t \cdot k \cdot \omega_{l_z}^\star) \\ R_z(\Delta t \cdot k \cdot \omega_{l_z}^\star) \end{bmatrix} \cdot {}^{l}\dot{q}^\star \tag{35} $$

with $R_z(\psi)$ the 3 by 3 rotation matrix by an angle ψ about the vertical axis. There is no rotation about the roll and pitch axes due to the assumption that the trunk is almost horizontal.

To get the reference position vector for all time steps of the prediction horizon an integration similar to the one that has been done for the prediction term of the footstep planner is performed. If $\omega_{l_z}^\star = 0$:

$$ \forall k \in [1, n_{steps}], \;\; q_k^\star = q_0 + k \; \Delta t \; {}^{l}\dot{q}^\star \tag{36} $$

If $\omega_{l_z^\star} \neq 0$:

$$x_k^\star = {}^lC_x + \frac{{}^l\dot{x}^\star \sin(\omega_{l_z}^\star\ k\ \Delta t) + {}^l\dot{y}^\star \left(\cos(\omega_{l_z}^\star\ k\ \Delta t) - 1\right)}{\omega_{l_z}^\star} \tag{37}$$

$$y_k^\star = {}^lC_y + \frac{-{}^l\dot{x}^\star \left(\cos(\omega_{l_z}^\star\ k\ \Delta t) - 1\right) + {}^l\dot{y}^\star \sin(\omega_{l_z}^\star\ k\ \Delta t)}{\omega_{l_z}^\star} \tag{38}$$

$$z_k^\star = {}^lC_z + k\ \Delta t\ {}^l\dot{z}^\star \tag{39}$$

$$\phi_k^\star = {}^l\phi + \frac{\omega_{l_x}^\star \sin(\omega_{l_z}^\star\ k\ \Delta t) + \omega_{l_y}^\star \left(\cos(\omega_{l_z}^\star\ k\ \Delta t) - 1\right)}{\omega_{l_z}^\star} \tag{40}$$

$$\theta_k^\star = {}^l\theta + \frac{-\omega_{l_x}^\star \left(\cos(\omega_{l_z}^\star\ k\ \Delta t) - 1\right) + \omega_{l_y}^\star \sin(\omega_{l_z}^\star\ k\ \Delta t)}{\omega_{l_z}^\star} \tag{41}$$

$$\psi_k^\star = 0 + k\ \Delta t\ \omega_{l_z}^\star \tag{42}$$

$$\tag{43}$$

Previous equations could be used in a general case for which there is a velocity control for all linear and angular components. However, in our case, since we want the quadruped to move around while keeping the trunk horizontal and at constant height, we want a velocity control in x, y and ψ and a position control in z, ϕ and θ to keep $\forall t,\ z(t) = h$ and $\phi(t) = \theta(t) = 0\ rad$. To avoid having too many feedback loop (reference velocity for z, ϕ and θ depending on the position error) we set $\forall k \in [1, n_{steps}]$:

$$\dot{z}_k^\star = 0 \text{ and } z_k^\star = h \tag{44}$$

$$\omega_{l_x}^\star = 0 \text{ and } \phi_k^\star = 0 \tag{45}$$

$$\omega_{l_y}^\star = 0 \text{ and } \theta_k^\star = 0 \tag{46}$$

To sum things up:

$$\forall k \in [1, n_{steps}],\ X_k^\star = \begin{bmatrix} {}^lC_x + \frac{{}^l\dot{x}^\star \sin(\omega_{l_z}^\star\ k\ \Delta t) + {}^l\dot{y}^\star \left(\cos(\omega_{l_z}^\star\ k\ \Delta t) - 1\right)}{\omega_{l_z}^\star} \\ {}^lC_y + \frac{-{}^l\dot{x}^\star \left(\cos(\omega_{l_z}^\star\ k\ \Delta t) - 1\right) + {}^l\dot{y}^\star \sin(\omega_{l_z}^\star\ k\ \Delta t)}{\omega_{l_z}^\star} \\ h \\ 0 \\ 0 \\ k\ \Delta t\ \omega_{l_z}^\star \\ {}^l\dot{x}^\star \cos(k\ \Delta t\ \omega_{l_z}^\star) - {}^l\dot{y}^\star \sin(k\ \Delta t\ \omega_{l_z}^\star) \\ {}^l\dot{x}^\star \sin(k\ \Delta t\ \omega_{l_z}^\star) + {}^l\dot{y}^\star \cos(k\ \Delta t\ \omega_{l_z}^\star) \\ 0 \\ 0 \\ 0 \\ \omega_{l_z}^\star \end{bmatrix} \tag{47}$$

The solver will work around the reference trajectory so we define the optimization state vector as follows:

$$\mathcal{X}_k = X_k - X_k^\star \tag{48}$$

6 Dynamics equations and constraints

The MPC works with a simple lumped mass model (centroidal dynamics). It can be written in world frame as follows:

$$m\ {}^{o}\ddot{C} = \sum_{i=0}^{n_c-1} {}^{o}f_i - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{49}$$

$$\frac{d}{dt}({}^{o}\mathcal{I}\,{}^{o}\omega) = \sum_{i=0}^{n_c-1} ({}^{o}r_i - {}^{o}C) \times {}^{o}f_i \tag{50}$$

With n_c the number of footholds, ${}^{o}f_i$ the reaction forces, ${}^{o}r_i$ the location of contact points, ${}^{o}C$ the position of the center of mass, ${}^{o}\mathcal{I}$ the rotational inertia tensor and ${}^{o}w$ the angular velocity of the body.

The first assumption is that roll and pitch angles are small, it follows that:

$$\begin{bmatrix} {}^{o}\dot{\phi} \\ {}^{o}\dot{\theta} \\ {}^{o}\dot{\psi} \end{bmatrix} \approx R_z(\psi)^{-1} \cdot {}^{o}\omega \tag{51}$$

$${}^{o}\mathcal{I} \approx R_z(\psi) \cdot {}^{b}\mathcal{I} \cdot R_z(\psi)^{-1} \tag{52}$$

The second assumption is that states are close to the desired trajectory so in equation 50 we can replace the position of the center of mass ${}^{o}C$ by the desired position for the center of mass.

The last assumption is that pitch and roll velocities are small so:

$$\frac{d}{dt}({}^{o}\mathcal{I}\,{}^{o}\omega) = {}^{o}\mathcal{I}\,{}^{o}\dot{\omega} + {}^{o}\omega \times ({}^{o}\mathcal{I}\,{}^{o}\omega) \approx {}^{o}\mathcal{I}\,{}^{o}\dot{\omega} \tag{53}$$

With these assumptions, equation 50 is simplified into:

$${}^{o}\mathcal{I}\ {}^{o}\dot{\omega} = \sum_{i=0}^{n_c-1} ({}^{o}r_i - {}^{o}C^{\star}) \times {}^{o}f_i \tag{54}$$

The local frame that the solver is working in is actually the world frame rotated by ψ about the vertical axis z so equation 51 becomes:

$$\begin{bmatrix} {}^{o}\dot{\phi} \\ {}^{o}\dot{\theta} \\ {}^{o}\dot{\psi} \end{bmatrix} \approx {}^{l}\omega \tag{55}$$

Equations 49 and 54 can be written in local frame:

$$m\ R_z(\psi)^{-1\,l}\ddot{C} = \sum_{i=0}^{n_c-1} R_z(\psi)^{-1\,l}f_i - R_z(\psi)^{-1} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{56}$$

$$R_z(\psi)^{-1\,l}\mathcal{I}R_z(\psi)\ R_z(\psi)^{-1\,l}\dot{\omega} = \sum_{i=0}^{n_c-1} R_z(\psi)^{-1}({}^{l}r_i - {}^{l}C^{\star}) \times R_z(\psi)^{-1\,l}f_i \tag{57}$$

As cross product is invariant by rotation these equations result in:

$$m\ {}^{l}\ddot{C} = \sum_{i=0}^{n_c-1} {}^{l}f_i - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{58}$$

$${}^{l}\mathcal{I}\ {}^{l}\dot{\omega} = \sum_{i=0}^{n_c-1} ({}^{l}r_i - {}^{l}C^{\star}) \times {}^{l}f_i \tag{59}$$

Once discretized and considering equation 55, evolution of state variables becomes $\forall k \in [0, n_{steps}-1]$:

$$\begin{bmatrix} {}^{l}x_{k+1} \\ {}^{l}y_{k+1} \\ {}^{l}z_{k+1} \end{bmatrix} = \begin{bmatrix} {}^{l}x_{k} \\ {}^{l}y_{k} \\ {}^{l}z_{k} \end{bmatrix} + \Delta t \begin{bmatrix} {}^{l}\dot{x}_{k} \\ {}^{l}\dot{y}_{k} \\ {}^{l}\dot{z}_{k} \end{bmatrix} \tag{60}$$

$$\begin{bmatrix} {}^{l}\phi_{k+1} \\ {}^{l}\theta_{k+1} \\ {}^{l}\psi_{k+1} \end{bmatrix} = \begin{bmatrix} {}^{l}\phi_{k} \\ {}^{l}\theta_{k} \\ {}^{l}\psi_{k} \end{bmatrix} + \Delta t \begin{bmatrix} \omega_{l_{x,k}} \\ \omega_{l_{y,k}} \\ \omega_{l_{z,k}} \end{bmatrix} \tag{61}$$

$$\begin{bmatrix} {}^{l}\dot{x}_{k+1} \\ {}^{l}\dot{y}_{k+1} \\ {}^{l}\dot{z}_{k+1} \end{bmatrix} = \begin{bmatrix} {}^{l}\dot{x}_{k} \\ {}^{l}\dot{y}_{k} \\ {}^{l}\dot{z}_{k} \end{bmatrix} + \Delta t \left(\sum_{i=0}^{n_{c,k}-1} \frac{{}^{l}f_{i,k}}{m} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right) \tag{62}$$

$$\begin{bmatrix} \omega_{l_{x,k+1}} \\ \omega_{l_{y,k+1}} \\ \omega_{l_{z,k+1}} \end{bmatrix} = \begin{bmatrix} \omega_{l_{x,k}} \\ \omega_{l_{y,k}} \\ \omega_{l_{z,k}} \end{bmatrix} + \Delta t \left({}^{l}\mathcal{I}^{-1} \sum_{i=0}^{n_{c,k}-1} [{}^{l}r_{i,k} - {}^{l}C_k^{\star}]_{\times} \cdot {}^{l}f_{i,k} \right) \tag{63}$$

In terms of constraints, friction cone conditions to avoid slipping are linearized to the first order:

$$\forall i \in [0,3], \forall k \in [0, n_{steps}-1],\ |f_{i,k}^{x}| \leq \mu\ f_{i,k}^{z} \text{ and } |f_{i,k}^{y}| \leq \mu\ f_{i,k}^{z} \tag{64}$$

An upper limit has to be set for contact forces to respect hardware limits (maximum torque of actuators). This limit is only applied to the z component since it will also limit the force along x and y due to the friction cone contraints.

$$\forall i \in [0,3], \forall k \in [0, n_{steps}-1],\ f_{i,k}^{z} \leq f_{max} \tag{65}$$

The quadruped cannot pull on the ground, it can only push so the normal component of the contact forces has to be positive:

$$\forall i \in [0,3], \forall k \in [0, n_{steps}-1], f_{i,k}^{z} \geq 0\ N \tag{66}$$

To be sure that there is no slipping, we could impose a minimal non-zero vertical component of the contact forces because if it is close to 0 N the friction cone is small so on the real robot slipping could happen. In practise to due the way the MPC is currently programmed to disable a foot when it is in swing phase we set a constraint that its contact force is equal to 0 so it is not directly compatible with $\forall i \in [0,3], \forall k \in [0, n_{steps}-1], f_{i,k}^{z} \geq f_{min}$. We would have to change more coefficients to temporarily disable this $f_{i,k}^{z} \geq f_{min}$ for feet in swing phase.

This minimal non-zero vertical component of the contact forces is taken into account by the inverse dynamics block (TSID) so even if the output of the MPC contains a 0 N vertical component for a foot in contact it will be equal to f_{min} after being processed by TSID.

7 MPC matrices for dynamics and constraints

Goal: create the matrices that are used by standard QP solvers. These solvers try to find a vector $\mathbb{X}$ that minimizes a cost function $\frac{1}{2}\mathbb{X}^T.P.\mathbb{X} + \mathbb{X}^T.Q$ under constraints $M.\mathbb{X} = N$ and $L.\mathbb{X} \leq K$. In this section the construction of matrices M, N, L and K is described.

The evolution of the state vector of the robot over time can be described as follows:

$$x(k+1) = A(k)x(k) + B(k)f(k) + g \tag{67}$$

Matrices A et B depends on k and $g = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ -9.81 \cdot \Delta t\ 0\ 0\ 0]^T$

The contact forces vector $f(k) = f_k$ always include the forces applied on the four feet even if some of them are not touching the ground. In that case we will set the problem in such a way that forces for such feet are not considered in the solving process.

$$f_k = [f_{0,k}^T \quad f_{1,k}^T \quad f_{2,k}^T \quad f_{3,k}^T]^T \tag{68}$$

$$\forall i \in [0,3],\ k \in [0, n_{steps}-1],\ f_{i,k} = \begin{bmatrix} f_{i,k}^x \\ f_{i,k}^y \\ f_{i,k}^z \end{bmatrix} \tag{69}$$

with $f_{i,k}^x$, $f_{i,k}^y$ and $f_{i,k}^z$ the components along the x, y and z axes of the solver frame for the i-th foothold at time step k.

Let's consider a case with only 3 time steps in the prediction horizon.

The goal of the MPC is to find contacts forces f that should be applied to have the state vector X of the robot as close as possible to $X^\star$. The QP solver outputs at the end of the optimization process the optimization vector $\mathbb{X}$ that minimizes the cost function locally (globally in the best case). The QP problem can be written in a simple way by putting both f (the output of the MPC) and $\mathcal{X}_k = X_k - X_k^\star$ (quantity that should be minimized) in the optimization vector:

$$\mathbb{X} = [\mathcal{X}_1^T \quad \mathcal{X}_2^T \quad \mathcal{X}_3^T \quad f_0^T \quad f_1^T \quad f_2^T]^T \tag{70}$$

Matrix M is defined as follows:

$$M = \begin{bmatrix} -I_{12} & 0_{12} & 0_{12} & B_0 & 0_{12} & 0_{12} \\ A_1 & -I_{12} & 0_{12} & 0_{12} & B_1 & 0_{12} \\ 0_{12} & A_2 & -I_{12} & 0_{12} & 0_{12} & B_2 \\ 0_{12} & 0_{12} & 0_{12} & E_0 & 0_{12} & 0_{12} \\ 0_{12} & 0_{12} & 0_{12} & 0_{12} & E_1 & 0_{12} \\ 0_{12} & 0_{12} & 0_{12} & 0_{12} & 0_{12} & E_2 \end{bmatrix} \tag{71}$$

A, B and E have a size of 12 by 12.

Matrix N is defined as follows:

$$N = \begin{bmatrix} -g \\ -g \\ -g \\ 0_{12\times 1} \\ 0_{12\times 1} \\ 0_{12\times 1} \end{bmatrix} + \begin{bmatrix} -A_0 X_0 \\ 0_{12\times 1} \\ 0_{12\times 1} \\ 0_{12\times 1} \\ 0_{12\times 1} \\ 0_{12\times 1} \end{bmatrix} + \begin{bmatrix} I_{12} & 0_{12} & 0_{12} & 0_{12} & 0_{12} & 0_{12} \\ -A_1 & I_{12} & 0_{12} & 0_{12} & 0_{12} & 0_{12} \\ 0_{12} & -A_2 & I_{12} & 0_{12} & 0_{12} & 0_{12} \end{bmatrix} \cdot \begin{bmatrix} X_1^\star \\ X_2^\star \\ X_3^\star \\ 0_{12\times 1} \\ 0_{12\times 1} \\ 0_{12\times 1} \end{bmatrix} \tag{72}$$

Matrix A_k for time step k is defined as follows:

$$A_k = \begin{bmatrix} I_3 & 0_3 & \Delta t \cdot I_3 & 0_3 \\ 0_3 & I_3 & 0_3 & \Delta t \cdot I_3 \\ 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix} \tag{73}$$

With the assumption that roll and pitch angles are small the inertia matrix of the robot in solver frame rotated according to the orientation of the robot at time step k is:

$${}^{l}\mathcal{I}_k = R_z(\Delta t \cdot k \cdot \omega_{l_z}^{\star}) \cdot {}^{b}\mathcal{I} \tag{74}$$

Matrix B_k for time step k is defined as follows:

$$B = \Delta t \cdot \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 \\ I_3/m & I_3/m & I_3/m & I_3/m \\ {}^{l}\mathcal{I}_k^{-1} \cdot [{}^{l}r_{0,k} - {}^{l}C_k^{\star}]_\times & {}^{l}\mathcal{I}_k^{-1} \cdot [{}^{l}r_{1,k} - {}^{l}C_k^{\star}]_\times & {}^{l}\mathcal{I}_k^{-1} \cdot [{}^{l}r_{2,k} - {}^{l}C_k^{\star}]_\times & {}^{l}\mathcal{I}_k^{-1} \cdot [{}^{l}r_{3,k} - {}^{l}C_k^{\star}]_\times \end{bmatrix} \tag{75}$$

with $({}^{l}r_{i,k} - {}^{l}C_k^{\star})$ the vector in local frame going from the desired position of the center of mass at time step k to the position of the i-th foothold. $[r_{k,i} - {}^{l}C_k^{\star}]_\times$ is the associated skew-symmetric matrix.

Matrix E_k for time step k is defined as follows:

$$E_k = \begin{bmatrix} e_{0,k} & 0_3 & 0_3 & 0_3 \\ 0_3 & e_{1,k} & 0_3 & 0_3 \\ 0_3 & 0_3 & e_{2,k} & 0_3 \\ 0_3 & 0_3 & 0_3 & e_{3,k} \end{bmatrix} \tag{76}$$

$e_{i,k} = 0_3$ if the i-th foot is touching the ground during time step k, $e_{i,k} = I_3$ otherwise. In fact, if $e_{i,k} = I_3$ then with $M.X = N$ we are setting the constraint that $f_{i,k} = [0 \;\; 0 \;\; 0]^T$ (no reaction force since the foot is not touching the ground).

Matrix L is defined as follows:

$$L = \begin{bmatrix} 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & F_\mu & 0_{20\times12} & 0_{20\times12} \\ 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & F_\mu & 0_{20\times12} \\ 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & F_\mu \end{bmatrix} \tag{77}$$

With:

$$F_\mu = \begin{bmatrix} G & 0_{5\times3} & 0_{5\times3} & 0_{5\times3} \\ 0_{5\times3} & G & 0_{5\times3} & 0_{5\times3} \\ 0_{5\times3} & 0_{5\times3} & G & 0_{5\times3} \\ 0_{5\times3} & 0_{5\times3} & 0_{5\times3} & G \end{bmatrix} \text{ and } G = \begin{bmatrix} 1 & 0 & -\mu \\ -1 & 0 & -\mu \\ 0 & 1 & -\mu \\ 0 & -1 & -\mu \\ 0 & 0 & -1 \end{bmatrix} \tag{78}$$

The K matrix is defined as $K = 0_{60\times1}$.

8 MPC cost function

QP solvers try to find a vector $\mathbb{X}$ that minimizes a cost function $\frac{1}{2}\mathbb{X}^T.P.\mathbb{X} + \mathbb{X}^T.Q$ under constraints $M.\mathbb{X} = N$ and $L.\mathbb{X} \leq K$. Matrices P and Q define the shape of the cost function.

The goal of the MPC is to find which contact forces should be applied at contact points so that the predicted trajectory of the center of mass is as close as possible to the reference trajectory. With previous notation, that means we want to minimize $|X - X^\star|$. Function $|\cdot|$ is not directly available with a matrix product $\frac{1}{2}\mathbb{X}^T.P.\mathbb{X} + \mathbb{X}^T.Q$ so we can try to minimize $(X - X^\star)^2$ instead. Since:

$$\mathbb{X} = \begin{bmatrix} \mathcal{X}_1 \\ \vdots \\ \mathcal{X}_{n_{steps}} \\ f_0 \\ \vdots \\ f_{n_{steps}-1} \end{bmatrix} = \begin{bmatrix} X_1 - X_1^\star \\ \vdots \\ X_{n_{steps}} - X_{n_{steps}}^\star \\ f_0 \\ \vdots \\ f_{n_{steps}-1} \end{bmatrix} \tag{79}$$

then the upper left portion of P can be diagonal:

$$P = \begin{bmatrix} c_{X,1} & & 0 & * \\ & \ddots & & * \\ 0 & & c_{X,n_{steps}} & * \\ * & * & * & * \end{bmatrix} \tag{80}$$

With $\forall k \in [1, n_{steps}]$, $c_{X,k}$ being 12 by 12 diagonal matrices with coefficients ≥ 0 the deviation from the reference trajectory is penalized by the cost function as it push the solver into minimizing the error $(X_k - X_k^\star)^2$. For safety reason, for energy consumption and for the actuators, it is better to keep the contact forces low if possible. That is why a small regularization term is added to slightly penalize the norm of contact forces. Since the $\sqrt{\cdot}$ function is not directly available in the matrix product of the cost function, we regularize the square of the norm instead $(\|f_k\|^2)$.

$$P = \begin{bmatrix} c_{X,1} & & 0 & * & * & * \\ & \ddots & & * & * & * \\ 0 & & c_{X,n_{steps}} & * & * & * \\ * & * & * & c_{f,0} & & 0 \\ * & * & * & & \ddots & \\ * & * & * & 0 & & c_{f,n_{steps}-1} \end{bmatrix} \tag{81}$$

With $\forall k \in [0, n_{steps} - 1]$, $c_{f,k}$ being 12 by 12 diagonal matrices with coefficients ≥ 0. There is no cross-coupling between $\mathcal{X}$ and force components so the upper-right and lower-left parts of P are zeros.

As there is no focus on any part of the prediction horizon, all $c_{X,k}$ are equal, same of all $c_{f,k}$. Coefficient at position $[i, i]$ in $c_{X,k}$ weights the deviation of the i-th component of the state vector from the reference trajectory. Remember that components of the state vector are in this order: $[{}^l\!x\ {}^l\!y\ {}^l\!z\ {}^l\!\phi\ {}^l\!\theta\ {}^l\!\psi\ {}^l\!\dot{x}\ {}^l\!\dot{y}\ {}^l\!\dot{z}\ \omega_{l_x}\ \omega_{l_y}\ \omega_{l_z}]$. Coefficient at position $[i, i]$ in $c_{f,k}$ weights the i-th component of the force vector for regularization purpose. Remember that components of the force vector are in this order: $[{}^lf_0^x\ {}^lf_0^y\ {}^lf_0^z\ {}^lf_1^x\ {}^lf_1^y\ {}^lf_1^z\ {}^lf_2^x\ {}^lf_2^y\ {}^lf_2^z\ v{}^lf_3^x\ {}^lf_3^y\ {}^lf_3^z]$. To regularize properly the norm

of contact forces $\|f_{i,k}\|^2 = (f_{i,k}^x)^2 + (f_{i,k}^y)^2 + (f_{i,k}^z)^2$ coefficients for the x, y and z components have to be equal:

$$\forall k \in [0, n_{steps} - 1],\ \forall i \in [0, 3],\ c_{f,k}[3i, 3i] = c_{f,k}[3i + 1, 3i + 1] = c_{f,k}[3i + 2, 3i + 2] \tag{82}$$

If no leg is privileged (to mimic a wounded leg we would try to apply less force with it) then all coefficients on the diagonal of $c_{X,k}$ are equal and $\forall k \in [0, n_{steps} - 1],\ c_{X,k} = w_f I_{12}$ with $w_f \in \mathbb{R}^+$

Matrix Q in $\mathbb{X}^T \cdot Q$ only contains zeroes since there is no reason to push $X_k - X_k^\star$ or f_k into being as negative/positive as possible. For instance if a coefficient of Q is positive then the solver will try to have the associated variable as negative as possible to have a high negative product between the coefficient and the variable since that minimizes the cost.

In the 3 time steps example of the previous section P has a size of 72 by 72 (12 x 3 lines/columns for $\mathcal{X}_{1,2,3}$ and 12 x 3 lines/columns for $f_{1,2,3}$) and Q has a size size 72 by 1.

The cost function during the optimization process is then:

$$cost(X - X^\star, f) = \sum_{k=1}^{n_{steps}} \left(\sum_{i=0}^{11} \left[w_X^i (X_k^i - X_k^{i,\star})^2 \right] + w_f \sum_{i=0}^{3} \left[(f_{i,k}^x)^2 + (f_{i,k}^y)^2 + (f_{i,k}^z)^2 \right] \right) \tag{83}$$

9 Output of the MPC

The desired reaction forces that need to be applied (by TSID or the real robot) are stored in f_0. It contains the desired reaction forces in local frame so they will have to be brought back to the world frame that TSID is working in.

The same applies for the next desired position of the robot that is stored in $\mathcal{X}_1$ and can be retrieved by adding $X_1^\star$ to $\mathcal{X}_1$. As the next position/orientation is expressed in local frame we would have to rotate them to be able to use them for the inverse dynamics.

10 Inverse dynamics

The goal of the inverse dynamics is to make the link between the high level control (MPC contact forces and desired position of footsteps) and the low level control (desired torques sent to the drivers of the actuators). To do that, we use Efficient Task Space Inverse Dynamics (TSID, `https://github.com/stack-of-tasks/tsid`). This library allows to perform task-orientated optimization-based inverse-dynamics control based on the rigid multi-body dynamics library Pinocchio. Unlike the MPC it does not consider a prediction horizon but only the current state of the robot and the current defined tasks.

Currently four kind of tasks are being used:

- Contact tasks for feet in contact with the ground to inform the solver that these feet should not move and that they can be used to apply forces on the ground.
- Force tasks to have the contact forces close to the desired contact forces outputted by the MPC. These tasks are associated with the contact tasks.

- Tracking tasks for feet in swing phase to follow the 3D trajectory generated by the foot trajectory generator to land at the position desired by the footstep planner.
- Posture task for all legs to get back to a default position if some degrees of freedom are not used.

One instance of the first three task is initially created and assigned to each foot. Then during the gait these tasks are enabled or disabled depending on the state of the foot. In swing phase only the tracking task is active while in stance phase only contact and force tasks are enabled. There exists a single posture task which is always active and affect the whole body.

A weight is assigned to each task to make it more or less important compared to the other ones. "Contact + Force" and 3D tracking are never active at the same time so they do not compete with each other. As the posture task is just intended to be use as a form of regularization, it should not interfere with the other tasks. Its weight is kept at least 10^{-2} times less than the others to mimic a hierarchical solver: the relative weight is so small that it does not impact the other tasks even if in practice all tasks are considered together during the solving process.

If the i-th foot is in stance phase then the force reference of its force task is updated with the desired contact force $[f_{i,0}^x \ f_{i,0}^y \ f_{i,0}^z]$ outputted by the MPC. If the i-th foot is in swing phase then its tracking task is updated with the desired position $[x^\star \ y^\star \ z^\star]$, velocity $[\dot{x}^\star \ \dot{y}^\star \ \dot{z}^\star]$ and acceleration $[\ddot{x}^\star \ \ddot{y}^\star \ \ddot{z}^\star]$ outputted by the foot trajectory generator associated with this foot.

The inverse dynamics solver is first updated with the current state of the quadruped (position/orientation/velocity of the base and angular position/velocity of the joints). It then tries to find the accelerations (base+joints) and the contact forces that minimize the cost function (weighted sum of task errors) while respecting the constraints (contacts, dynamics equations, torque limits). Joint torques can be retrieved at the end of the optimization thanks to the accelerations and the contact forces.

Retrieved torques are sent to the PyBullet simulator that makes the assumption that torques are perfectly followed by the actuators. On the real robot they would be sent to the drivers that would have to modulate the current sent to the actuators to get the desired torques.