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.

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. 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 ${}^{1}T_2$ and the rotation part ${}^{1}R_2$ 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 with the altitude of a foot being the z coordinate of its center. 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 ${}^bV$ and ${}^bW$ in base frame:

$$\begin{aligned} {}^bV &= ({}^oR_b)^{-1} \cdot {}^oV && (9)\\ {}^bW &= ({}^oR_b)^{-1} \cdot {}^oW && (10) \end{aligned}$$

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, 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 ${}^ofeet$ are directly retrieved from PyBullet and transformed in local frame for the MPC:

$${}^lfeet = ({}^oM_l)^{-1} \cdot {}^ofeet \tag{17}$$

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 two 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:

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

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

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

A symmetry term is then 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 but 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 the symmetry term tries to land $\frac{t_{stance}}{2}v$ in front of the shoulder. That way, feet in contact spend half the stance phase in front of the shoulder and the other half behind it.

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

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 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 velocity.

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

The feedback gain k is equal to 0.03.

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 than what the feedback term does for the linear velocity.

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

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 $x_0 + \Delta x + t_{swing}\ \dot{x}$ due to the movement of the base. With the assumption that the current and reference velocities do not change much over one period of gait then feet can 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 ${}^{l}\dot{\psi} \neq 0$:

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

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

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

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

If ${}^{l}\dot{\psi} = 0$:

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

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

The remaining duration t_r for the swing phase of a foot can be directly retrieved using information contained in the gait matrix: 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 the second row of the $gait(0)$ matrix of equation 18 would be:

$$[\ 7\ \ r_{x,0}\ \ r_{y,0}\ \ 0\ \ NaN\ \ NaN\ \ NaN\ \ NaN\ \ NaN\ \ NaN\ \ r_{x,3}\ \ r_{y,3}\ \ 0\ ] \tag{30}$$

4 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.

$$ {}^l\dot{q}^\star = [{}^l\dot{x}^\star \;\; {}^l\dot{y}^\star \;\; {}^l\dot{z}^\star \;\; {}^l\dot{\phi}^\star \;\; {}^l\dot{\theta}^\star \;\; {}^l\dot{\psi}^\star]^T \tag{31} $$

The velocity vector of the robot is:

$$ {}^l\dot{q} = [{}^l\dot{x} \;\; {}^l\dot{y} \;\; {}^l\dot{z} \;\; {}^l\dot{\phi} \;\; {}^l\dot{\theta} \;\; {}^l\dot{\psi}]^T \tag{32} $$

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 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. As the frame the solver is working in is the local frame, initial conditions of the solving process are as follows:

$$ q_0 = [{}^lx \;\; {}^ly \;\; {}^lz \;\; {}^l\phi \;\; {}^l\theta \;\; {}^l\psi]^T = [{}^lC_x \;\; {}^lC_y \;\; {}^lC_z \;\; {}^l\phi \;\; {}^l\theta \;\; 0]^T \tag{33} $$

$$ \dot{q}_0 = [{}^l\dot{x} \;\; {}^l\dot{y} \;\; {}^l\dot{z} \;\; {}^l\dot{\phi} \;\; {}^l\dot{\theta} \;\; {}^l\dot{\psi}]^T \tag{34} $$

With $[{}^lC_x \;\; {}^lC_y \;\; {}^lC_z]$ the position of the center of mass in local frame.

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

$$ X = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \qquad X^\star = \begin{bmatrix} q^\star \\ \dot{q}^\star \end{bmatrix} \tag{35} $$

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:

$$ \forall k \in [1, N], \;\; \dot{q}^\star_k = \begin{bmatrix} R_z(\Delta t \cdot k \cdot \dot{\psi}^\star) \\ R_z(\Delta t \cdot k \cdot \dot{\psi}^\star) \end{bmatrix} \cdot {}^l\dot{q}^\star \tag{36} $$

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 ${}^l\dot{\psi} = 0$:

$$ \forall k \in [1, N], \;\; q^\star_k = q_0 + k \;\Delta t \; {}^l\dot{q}^\star \tag{37} $$

If ${}^{l}\dot{\psi} \neq 0$:

$$x_k^\star = {}^{l}C_x + \frac{{}^{l}\dot{x}^\star \sin({}^{l}\dot{\psi}^\star \ k \ \Delta t) + {}^{l}\dot{y}^\star \Big(\cos({}^{l}\dot{\psi}^\star \ k \ \Delta t) - 1\Big)}{{}^{l}\dot{\psi}^\star} \tag{38}$$

$$y_k^\star = {}^{l}C_y + \frac{-{}^{l}\dot{x}^\star \Big(\cos({}^{l}\dot{\psi}^\star \ k \ \Delta t) - 1\Big) + {}^{l}\dot{y}^\star \sin({}^{l}\dot{\psi}^\star \ k \ \Delta t)}{{}^{l}\dot{\psi}^\star} \tag{39}$$

$$z_k^\star = {}^{l}C_z + k \ \Delta t \ {}^{l}\dot{z}^\star \tag{40}$$

$$\phi_k^\star = {}^{l}\phi + \frac{{}^{l}\dot{\phi}^\star \sin({}^{l}\dot{\psi}^\star \ k \ \Delta t) + {}^{l}\dot{\theta}^\star \Big(\cos({}^{l}\dot{\psi}^\star \ k \ \Delta t) - 1\Big)}{{}^{l}\dot{\psi}^\star} \tag{41}$$

$$\theta_k^\star = {}^{l}\theta + \frac{-{}^{l}\dot{\phi}^\star \Big(\cos({}^{l}\dot{\psi}^\star \ k \ \Delta t) - 1\Big) + {}^{l}\dot{\theta}^\star \sin({}^{l}\dot{\psi}^\star \ k \ \Delta t)}{{}^{l}\dot{\psi}^\star} \tag{42}$$

$$\psi_k^\star = 0 + k \ \Delta t \ {}^{l}\dot{\psi}^\star \tag{43}$$

$$\tag{44}$$

Previous equations can be used a general case in 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 $\forall k \in [1, N]$:

$$\dot{z}_k^\star = 0 \text{ and } z_k^\star = h \tag{45}$$

$$\dot{\phi}_k^\star = 0 \text{ and } \phi_k^\star = 0 \tag{46}$$

$$\dot{\theta}_k^\star = 0 \text{ and } \theta_k^\star = 0 \tag{47}$$

To sum things up:

$$\forall k \in [1, N],\ X_k^\star = \begin{bmatrix} {}^{l}C_x + \frac{{}^{l}\dot{x}^\star \sin({}^{l}\dot{\psi}^\star \ k \ \Delta t) + {}^{l}\dot{y}^\star \left(\cos({}^{l}\dot{\psi}^\star \ k \ \Delta t) - 1\right)}{{}^{l}\dot{\psi}^\star} \\ {}^{l}C_y + \frac{-{}^{l}\dot{x}^\star \left(\cos({}^{l}\dot{\psi}^\star \ k \ \Delta t) - 1\right) + {}^{l}\dot{y}^\star \sin({}^{l}\dot{\psi}^\star \ k \ \Delta t)}{{}^{l}\dot{\psi}^\star} \\ h \\ 0 \\ 0 \\ k \ \Delta t \ {}^{l}\dot{\psi}^\star \\ {}^{l}\dot{x}^\star \cos(k \ \Delta t \ {}^{l}\dot{\psi}^\star) - {}^{l}\dot{y}^\star \sin(k \ \Delta t \ {}^{l}\dot{\psi}^\star) \\ {}^{l}\dot{x}^\star \sin(k \ \Delta t \ {}^{l}\dot{\psi}^\star) + {}^{l}\dot{y}^\star \cos(k \ \Delta t \ {}^{l}\dot{\psi}^\star) \\ 0 \\ 0 \\ 0 \\ {}^{l}\dot{\psi}^\star \end{bmatrix} \tag{48}$$

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

$$\mathcal{X}_k = X_k - X_k^\star \tag{49}$$

5 Dynamics equations and constraints

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

$$m\ {}^{o}\ddot{C} = \sum_{i=0}^{n_c-1} {}^{o}f_i - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{50}$$

$$\frac{d}{dt}({}^{o}I\,{}^{o}\omega) = \sum_{i=0}^{n_c-1} ({}^{o}r_i - {}^{o}C) \times {}^{o}f_i \tag{51}$$

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, ${}^{o}I$ 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:

$$\begin{bmatrix} {}^{o}\dot{\phi} \\ {}^{o}\dot{\theta} \\ {}^{o}\dot{\psi} \end{bmatrix} \approx R_z(\psi)^{-1} \cdot {}^{o}\omega \tag{52}$$

$${}^{o}I \approx R_z(\psi) \cdot {}^{b}I \cdot R_z(\psi)^{-1} \tag{53}$$

The second assumption is that states are close to the desired trajectory so in equation 51 we can replace the actual position of the center of mass ${}^{o}C$ by the desired trajectory for the center of mass.

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

$$\frac{d}{dt}({}^{o}I\,{}^{o}\omega) = {}^{o}I\,{}^{o}\dot{\omega} + {}^{o}\omega \times ({}^{o}I\,{}^{o}\omega) \approx {}^{o}I\,{}^{o}\dot{\omega} \tag{54}$$

With these assumptions, equation 51 is simplified into:

$${}^{o}I\ {}^{o}\dot{\omega} = \sum_{i=0}^{n_c-1} ({}^{o}r_i - {}^{o}C^{\star}) \times {}^{o}f_i \tag{55}$$

The local frame that the solver is working in is actually the world frame rotated by ψ about the vertical axis z so equation 52 becomes:

$$\begin{bmatrix} {}^{o}\dot{\phi} \\ {}^{o}\dot{\theta} \\ {}^{o}\dot{\psi} \end{bmatrix} \approx {}^{l}\omega \tag{56}$$

Equations 50 and 55 can be written in local frame:

$$m\ R_z(\psi)^{-1\,l}\ddot{C} = \sum_{i=0}^{n_c-1} R_z(\psi)^{-1\,l}f_i - R_z(\psi)^{-1} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{57}$$

$$R_z(\psi)^{-1\,l}I R_z(\psi)\ R_z(\psi)^{-1\,l}\dot{\omega} = \sum_{i=0}^{n_c-1} R_z(\psi)^{-1}({}^{l}r_i - {}^{l}C^{\star}) \times R_z(\psi)^{-1\,l}f_i \tag{58}$$

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

$$m\ {}^{l}\ddot{C} = \sum_{i=0}^{n_c-1} {}^{l}f_i - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \tag{59}$$

$${}^{l}I\ {}^{l}\dot{\omega} = \sum_{i=0}^{n_c-1} ({}^{l}r_i - {}^{l}C^{\star}) \times {}^{l}f_i \tag{60}$$

Once discretized, evolution of state variables becomes, $\forall k \in [0, N-1]$:

$$\begin{bmatrix} {}^{l}x_{k+1} \\ {}^{l}y_{k+1} \\ {}^{l}z_{k+1} \end{bmatrix} = \begin{bmatrix} {}^{l}x_{k} \\ {}^{l}y_{k} \\ {}^{l}z_{k} \end{bmatrix} + \Delta t \begin{bmatrix} {}^{l}\dot{x}_{k} \\ {}^{l}\dot{y}_{k} \\ {}^{l}\dot{z}_{k} \end{bmatrix} \tag{61}$$

$$\begin{bmatrix} {}^{l}\phi_{k+1} \\ {}^{l}\theta_{k+1} \\ {}^{l}\psi_{k+1} \end{bmatrix} = \begin{bmatrix} {}^{l}\phi_{k} \\ {}^{l}\theta_{k} \\ {}^{l}\psi_{k} \end{bmatrix} + \Delta t \begin{bmatrix} \omega_{{}^{l}x,k} \\ \omega_{{}^{l}y,k} \\ \omega_{{}^{l}z,k} \end{bmatrix} \tag{62}$$

$$\begin{bmatrix} {}^{l}\dot{x}_{k+1} \\ {}^{l}\dot{y}_{k+1} \\ {}^{l}\dot{z}_{k+1} \end{bmatrix} = \begin{bmatrix} {}^{l}\dot{x}_{k} \\ {}^{l}\dot{y}_{k} \\ {}^{l}\dot{z}_{k} \end{bmatrix} + \Delta t \left(\sum_{i=0}^{n_{c,k}-1} \frac{{}^{l}f_{i,k}}{m} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \right) \tag{63}$$

$$\begin{bmatrix} \omega_{{}^{l}x,k+1} \\ \omega_{{}^{l}y,k+1} \\ \omega_{{}^{l}z,k+1} \end{bmatrix} = \begin{bmatrix} \omega_{{}^{l}x,k} \\ \omega_{{}^{l}y,k} \\ \omega_{{}^{l}z,k} \end{bmatrix} + \Delta t \left({}^{l}I^{-1} \sum_{i=0}^{n_{c,k}-1} [{}^{l}r_{i,k} - {}^{l}C_k^{\star}]_{\times} \cdot {}^{l}f_{i,k} \right) \tag{64}$$

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

$$\forall i \in [0,3], \forall k \in [0, N-1],\ |f_{i,k,x}| \leq \mu\ f_{i,k,z} \text{ and } |f_{i,k,y}| \leq \mu\ f_{i,k,z} \tag{65}$$

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.

$$\forall i \in [0,3], \forall k \in [0, N-1],\ f_{i,k,z} \leq f_{max} \tag{66}$$

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

$$\forall i \in [0,3], \forall k \in [0, N-1], f_{i,k,z} \geq 0\ N \tag{67}$$

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 $\forall i \in [0,3], \forall k \in [0, N-1], f_{i,k,z} \geq f_{min}$. 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.

6 Solver matrices

Goal: create the matrices that are used by standard QP solvers. Those solvers tries to find the vector $\mathbb{X}$ that minimizes $\frac{1}{2}\mathbb{X}^T.P.\mathbb{X} + \mathbb{X}^T.Q$ under constraints $M.\mathbb{X} = N$ and $L.\mathbb{X} \leq K$.

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

$$x(k+1) = A(k)x(k) + B(k)f(k) + g \tag{68}$$

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

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 at all in the solving process.

$$f_k = [f_{k,0}^T \quad f_{k,1}^T \quad f_{k,2}^T \quad f_{k,3}^T]^T \tag{69}$$

$$f_{k,i} = \begin{bmatrix} f_{k,i}^x \\ f_{k,i}^y \\ f_{k,i}^z \end{bmatrix} \tag{70}$$

with $f_{k,i}^x$, $f_{k,i}^y$ and $f_{k,i}^z$ the components about 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.

Since we are optimizing both the state vector X of the robot to have it close to $X^\star$ and the contacts forces f which are the output of the MPC, the solver vector $\mathbb{X}$ is then:

$$\mathbb{X} = [\mathcal{X}_1 \quad \mathcal{X}_2 \quad \mathcal{X}_3 \quad f_0 \quad f_1 \quad f_2]^T \tag{71}$$

Matrix M is defined as follows:

$$M = \begin{bmatrix} -I_{12} & 0_{12} & 0_{12} & B_0 & 0_{12} & 0_{12} \\ A_1 & -I_{12} & 0_{12} & 0_{12} & B_1 & 0_{12} \\ 0_{12} & A_2 & -I_{12} & 0_{12} & 0_{12} & B_2 \\ 0_{12} & 0_{12} & 0_{12} & E_0 & 0_{12} & 0_{12} \\ 0_{12} & 0_{12} & 0_{12} & 0_{12} & E_1 & 0_{12} \\ 0_{12} & 0_{12} & 0_{12} & 0_{12} & 0_{12} & E_2 \end{bmatrix} \tag{72}$$

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

Matrix N is defined as follows:

$$N = \begin{bmatrix} -g \\ -g \\ -g \\ 0_{12\times1} \\ 0_{12\times1} \\ 0_{12\times1} \end{bmatrix} + \begin{bmatrix} -A_0X_0 \\ 0_{12\times1} \\ 0_{12\times1} \\ 0_{12\times1} \\ 0_{12\times1} \\ 0_{12\times1} \end{bmatrix} + \begin{bmatrix} I_{12} & 0_{12} & 0_{12} & 0_{12} & 0_{12} & 0_{12} \\ -A_1 & I_{12} & 0_{12} & 0_{12} & 0_{12} & 0_{12} \\ 0_{12} & -A_2 & I_{12} & 0_{12} & 0_{12} & 0_{12} \end{bmatrix} \cdot \begin{bmatrix} X_1^\star \\ X_2^\star \\ X_3^\star \\ 0_{12\times1} \\ 0_{12\times1} \\ 0_{12\times1} \end{bmatrix} \tag{73}$$

Matrix A_k for time step k is defined as follows:

$$A = \begin{bmatrix} I_3 & 0_3 & \Delta t \cdot I_3 & 0_3 \\ 0_3 & I_3 & 0_3 & \Delta t \cdot I_3 \\ 0_3 & 0_3 & I_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & I_3 \end{bmatrix} \tag{74}$$

The inertia matrix of the robot in solver frame rotated according to the orientation of the robot at time step k is:

$$\mathcal{I}_k = R_z(\Delta t \cdot k \cdot \dot{\phi}^\star) \cdot \mathcal{I} \tag{75}$$

Matrix B_k for time step k is defined as follows:

$$B = \begin{bmatrix} 0_3 & 0_3 & 0_3 & 0_3 \\ 0_3 & 0_3 & 0_3 & 0_3 \\ \Delta t/m \cdot I_3 & \Delta t/m \cdot I_3 & \Delta t/m \cdot I_3 & \Delta t/m \cdot I_3 \\ \Delta t \cdot \mathcal{I}_k^{-1} \cdot [r_{k,0}]_\times & \Delta t \cdot \mathcal{I}_k^{-1} \cdot [r_{k,1}]_\times & \Delta t \cdot \mathcal{I}_k^{-1} \cdot [r_{k,2}]_\times & \Delta t \cdot \mathcal{I}_k^{-1} \cdot [r_{k,3}]_\times \end{bmatrix} \tag{76}$$

with $r_{k,i}$ the vector expressed in solver frame going from the position of the robot at time step k to the position of the i-th foothold and $[r_{k,i}]_\times$ the associated skew-symmetric matrix.

Matrix E_k for time step k is defined as follows:

$$E_k = \begin{bmatrix} e_{k,0} & 0_3 & 0_3 & 0_3 \\ 0_3 & e_{k,1} & 0_3 & 0_3 \\ 0_3 & 0_3 & e_{k,2} & 0_3 \\ 0_3 & 0_3 & 0_3 & e_{k,3} \end{bmatrix} \tag{77}$$

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

Matrix L is defined as follows:

$$L = \begin{bmatrix} 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & F_\mu & 0_{20\times12} & 0_{20\times12} \\ 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & F_\mu & 0_{20\times12} \\ 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & 0_{20\times12} & F_\mu \end{bmatrix} \tag{78}$$

With:

$$F_\mu = \begin{bmatrix} C & 0_{5\times3} & 0_{5\times3} & 0_{5\times3} \\ 0_{5\times3} & C & 0_{5\times3} & 0_{5\times3} \\ 0_{5\times3} & 0_{5\times3} & C & 0_{5\times3} \\ 0_{5\times3} & 0_{5\times3} & 0_{5\times3} & C \end{bmatrix} \text{ and } C = \begin{bmatrix} 1 & 0 & -\mu \\ -1 & 0 & -\mu \\ 0 & 1 & -\mu \\ 0 & -1 & -\mu \\ 0 & 0 & -1 \end{bmatrix} \tag{79}$$

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

Matrix P in the cost function $\frac{1}{2}\mathbb{X}^T.P.\mathbb{X} + \mathbb{X}^T.Q$ is diagonal. That way the first term of the cost function looks like $\sum_{k=1}^{3} c_{k,X} \cdot (X_1 - X_1^\star)^2 + \sum_{k=0}^{2} c_{k,f} \cdot f_k^2$ with $c_{k,X}$ and $c_{k,f}$ with size $(12, 1)$. With $c_{k,X} > 0$ it push the solver into minimizing the error $X_k - X_k^\star$ to the reference trajectory that we want to follow while $c_{k,f}$ are small positive coefficients that act as a regularization of the force to get a solution with a norm as low as possible for the reaction force. In the 3 time steps example 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 $f_{1,2,3}$).

Matrix Q involved in $\mathbb{X}^T.Q$ 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 which minimizes the cost. In the 3 time steps example Q will be of size 72 by 1.

7 Output of the MPC

The desired reaction forces that need to be applied (by TSID or the real robot) are stored in f_0. Since the solver frame and local frame are initially superposed then f_0 directly contains the desired reaction forces in the local frame of the robot. To be used in TSID, these forces need to be rotated to be brought back in TSID's world frame.

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 the solver/local frame, it needs to be rotated to be brought back in TSID's world frame.