MPC based Re-planning for Quadruped Robots

Michele Focchi* Romeo Orsolino* Claudio Semini*

I. Introduction

Re-planning is an important feature to overcome accumulation of errors, intended as differences between *desired* and *actual* trajectories. These errors can be due to tracking inaccuracies, controller delays, simplified/inaccurate models or external disturbances and are very common in legged robots where interaction forces with the ground are always present. An Model Predictive Control (MPC)-based re-planning has been originally implemented for bipeds in [1], [2] to accommodate strong disturbances and more recently for quadrupeds in [3]. All these works are still restricted to flat terrain. Moreover, there is a limited study on the influence of the size of the re-planning window on the locomotion performances. *Concept of re-planning*. Online re-planning is crucial legged robots that are meant to traverse rough environments, because it represents a mechanism to adapt to the terrain (terrain adaptation), quickly recover from planning errors and (simultaneously) accommodate for user velocity set-point, visual inputs, terrain change and external impacts. In section V we chose an MPC implementation for our re-planning strategy. The planning horizon should be set just large enough to realize the necessary *anticipative* body motions for the future foothold locations. On the other hand, the re-planning phase should be frequent enough to mitigate the accumulation of errors. Finally, the discretization of the trajectory must be dense enough to be able to capture the main robot dynamics, but not too much because the problem size would increase, slowing down the optimization and making impossible to be carried out in an *online* fashion.

II. Model

As a model we use the well known Linear Inverted Pendulum (LIP). With the usual assumption that the Center of Mass (CoM) does not move vertically. This provides two decoupled linear equations for the horizontal dynamics (e.g. for X and Y directions) [1] that link the CoM x_c to the Zero Moment Point (ZMP) z:

$$z = x_c - \frac{h_{com}}{g}\ddot{x}_c \tag{1}$$

where $x_c = \begin{bmatrix} x & y \end{bmatrix}^T, z = \begin{bmatrix} z_x & z_y \end{bmatrix}^T \in \mathbb{R}^2$ and h_{com} is the CoM height w.r.t the ground. It is known that a LIP model is an oversimplification of the dynamics of the robot. However, despite the discrepancy with the real dynamics, we noted that re-planning every step was sufficient to keep the errors small. Henceforth,*for this section*, we consider only the X coordinate, being the development for the Y coordinate equivalent. We discretized the trajectories for both the CoM and the ZMP along the horizon with N piece-wise cubic polynomials with constant jerk $\dddot{x} \in \mathbb{R}$ over time intervals of constant lengths ΔT. Then the state of the system a time $t = k\Delta T$ with $k = 1,...,N$ is: $\hat{x}_k = \begin{bmatrix} x(kT) & \dot{x}(k\Delta T) & \ddot{x}(k\Delta T) \end{bmatrix}^T$ with input $\dddot{x}_k = \dddot{x}(k\Delta T)$ and output $z_k = z(k\Delta T)$. integrating the constant jerk over the time intervals leads to the following recursive relationship:

$$\begin{cases} \ddot{x}_{k+1} = \ddot{x}_k + \dddot{x}_k\Delta T \\ \dot{x}_{k+1} = \dot{x}_k + \ddot{x}_k\Delta T + \dddot{x}_k\Delta T^2/2 \\ x_{k+1} = x_k + \dot{x}_k\Delta T + \ddot{x}_k\Delta T^2/2 + \dddot{x}_k\Delta T^3/6 \end{cases} \tag{2}$$

that can be rewritten in matrix form as:

$$\begin{aligned} \hat{x}_{k+1} &= A\hat{x}_k + B\dddot{x}_k \\ z_{x_k} &= C_z\hat{x}_k \end{aligned} \tag{3}$$

with:

$$A = \begin{bmatrix} 1 & \Delta T & \Delta T^2/2 \\ 0 & 1 & \Delta T \\ 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} \Delta T^3/6 \\ \Delta T^2/2 \\ \Delta T \end{bmatrix}$$

$$C_z = \begin{bmatrix} 1 & 0 & -\frac{h_{com}}{g} \end{bmatrix}$$

III. Analytic formulation

Iterating the relation (3) N times, we can relate, at once, N values of the jerk with N values of the CoM state:

$$\begin{bmatrix} \hat{x}_2 \\ \hat{x}_3 \\ \vdots \\ \hat{x}_N \end{bmatrix} = \begin{bmatrix} A \\ AA \\ \vdots \\ A^{N-1} \end{bmatrix} x_1 + \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-2}B & A^{N-3}B & \dots & B \end{bmatrix} \begin{bmatrix} \dddot{x}_1 \\ \dddot{x}_2 \\ \vdots \\ \dddot{x}_N \end{bmatrix} \tag{4}$$

In a similar fashion, exploiting the ZMP equation (1), we can predict the whole ZMP trajectory $Z_x \in \mathbb{R}^N$ with: [2]

$$Z_x = Z_0\hat{x}_1 + Z_u\dddot{X} \tag{5}$$

where $Z_0 \in \mathbb{R}^{N\times 3}$ and $Z_u \in \mathbb{R}^{N\times N}$.

IV. Optimization problem

The optimization problem we use for planning the CoM trajectory is the following quadratic program:

$$\begin{aligned} \min_{\substack{\dddot{X} = \dddot{x}_1,...,\dddot{x}_N \\ \dddot{Y} = \dddot{y}_1,...,\dddot{y}_N}} \quad & \frac{1}{2}R\sum_{i=1}^{N} \dddot{x}_i^2 + \dddot{y}_i^2 \\ \text{s.t.} \quad & A_pZ + b_p \geq 0 \end{aligned} \tag{6}$$

*Department of Advanced Robotics, Istituto Italiano di Tecnologia, Genova, Italy. *email*:{ michele.focchi, romeo.orsolino, claudio.semini}@iit.it.

[1]In the case of quadrupeds, the optimization problem is not decoupled anymore if stance polygon constraints are added, because they are formed by half-spaces that are both dependent on X and Y coordinates.

[2]Note that the capital letters for the variables z_x and x, $\dddot{x}$ represent arrays for the whole trajectory, as in [2].

The decision variables are the jerk trajectories, *henceforth we consider both X and Y components*. To achieve smoothness we minimize the square of the 2-norm of the jerk, while we set (linear) hard constraints on the ZMP trajectory $Z = \begin{bmatrix} Z_x & Z_y \end{bmatrix}^T$. The goal is to have the ZMP always inside the set of support polygons, to ensure *dynamic* stability. Matrix A_p and b_p encode the constraints (e.g. half-spaces) of the support polygon active edges for each sample of the trajectory. Exploiting (5) these constraints can be expressed as a function of the jerk:

$$\underbrace{A_p \begin{bmatrix} Z_u & 0 \\ 0 & Z_u \end{bmatrix}}_{\tilde{A}} \begin{bmatrix} \dddot{X} \\ \dddot{Y} \end{bmatrix} + \underbrace{b_p + A_p \begin{bmatrix} Z_0 & 0 \\ 0 & Z_0 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{y}_1 \end{bmatrix}}_{\tilde{b}} \geq 0 \qquad (7)$$

V. RE-PLANNING ALGORITHM

According to the MPC idea, we aim to re-plan from the *actual* state of the CoM and of the feet, that will serve as initial state for the next optimization. The procedure to follow for re-planning is the following:

Algorithm 1 Replanning

```
1: while 1 do
2:     if (sample % replWindow) == 0 then        ▷ true when repl. time is elapsed
3:         replanningStage ++
4:         currSwing ← gaitSchedule.next()
5:         initialFeet ← actualFeet
6:         initialCoM ← actualCoM
7:         footHolds ← compFootholds($\begin{bmatrix} \dot{X}^{ref} \\ \dot{Y}^{ref} \end{bmatrix}$,initialFeet, initialCoM, currSwing, N)
8:         $\tilde{A}$,$\tilde{b}$ ← buildConstraintMatrix(footHolds)
9:         $\dddot{X}, \dddot{Y}$ ← soveQP(initialCoM, $\tilde{A}, \tilde{b}$)
10:        sampleW ← 0
11:    else
12:        sampleW ++ ▷ sampleW records the index inside the replanning window
13:    end if
14:    desiredCoM ← update(desiredCoM, $\dddot{X}(sampleW)$, $\dddot{Y}(sampleW)$)
15: end while
```

where $\begin{bmatrix} \dot{X}^{ref} & \dot{Y}^{ref} \end{bmatrix}^T$ is the desired velocity set by the user. After the first plan computation, the robot starts to realize the plan sample by sample following the desired CoM state. When the re-planning time is elapsed (e.g. last sample of the re-planning window), the foot-holds are recomputed (function *compFootholds*) to build a new constraint matrix followed by a new optimization.

A. Haptic mode

To simplify the synchronization among the consecutive optimizations, we decided to perform the re-planning at the *touchdown* event. Indeed, we believe that the touch-down events are relevant moments to re-plan because the stance state of the robot changes and there are interaction forces from the ground. This means the re-planning window will depend on the *touchdown* event that will be *haptic* (e.g. we continue the trajectory until a stable contact is achieved checking the ground reaction forces). This *haptic mode* has beneficial effect on locomotion because it enables to adapt to the real terrain feature rather than blindly enforce the planned trajectory [4], thus preventing the accumulation of errors.

B. Foothold correction

Now, let us imagine an external push moves the CoM away from the desired trajectory, in a way that it gets very close to the support polygon edges. According to the intensity of the push, even if the trajectory remains feasible, the robustness can be dramatically reduced in the subsequent steps (see this video) unless the whole set of footholds is recomputed about the new position of the robot. In essence, the foothold correction it consists in recomputing (for each leg) the first foothold in the horizon about the *actual* position of the hip rather than with respect to the previous foot position. This allows to avoid progressive degeneration of the support polygon.

VI. PRELIMINARY RESULTS

In a second video [3] we show preliminary simulations of the proposed approach with HyQ walking on flat terrain. We wish to demonstrate how the re-planning capability of our planner is able to: 1) reflect promptly a change of desired velocity coming from the user, 2) accommodate moderate lateral disturbances. This work represents a step forward with respect to our previous static heuristic planner [4] in several ways: 1) we were able to double the locomotion speed, 2) the trunk can move while swinging the leg (in [4] the trunk kept still during the swing), 3) we achieved *dynamic* rather than *static* stability and significantly reduced the backward motions. We set the *horizon* to 8 *s* with a prediction of 8 footholds, while we set the *re-planning window* to 1 touch-down event. To achieve *online* feasibility of the optimization, we discretized the trajectory with a time resolution of 40 *ms* and performed interpolation to accommodate the desired CoM to the higher rate of the planner (250 *Hz*). With this parametrization the *QP* was solved in an average of 58 *ms* with a standard deviation of 6 *ms*. The video also shows how the *foothold correction* prevents the degeneration of the support polygon. In another simulation we show that, if the *re-planning window* is increased to 2 touch-down events, the influence of the modeling errors (due to the use of a simplified dynamics in the optimization), it becomes too large. Another simulation shows what happens when the *haptic mode* is disabled, (e.g. touchdown is commanded by the planner). In this case, the discrepancy between the real touch-down and the predicted one, introduces disturbance forces that affect the robot pitch. Finally, we show the effect of adding an additional term in the cost function to track the desired velocity resulting in a faster motion.

REFERENCES

[1] P.-B. Wieber, "Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations," *IEEE-RAS International Conference on Humanoid Robots*, 2006.

[2] H. Diedam, D. Dimitrov, P.-b. Wieber, K. Mombaur, M. D. Online, and M. Diehl, "Online Walking Gait Generation with Adaptive Foot Positioning through Linear Model Predictive Control To cite this version : Online Walking Gait Generation with Adaptive Foot Positioning through Linear Model Predictive Control," pp. 22–26, 2009.

[3] C. D. Bellicoso, F. Jenelten, C. Gehring, and M. Hutter, "Dynamic Locomotion through Online Nonlinear Motion Optimization for Quadrupedal Robots," *IEEE Robotics and Automation Letters*, vol. 3766, no. c, pp. 1–1, 2018.

[4] M. Focchi, R. Orsolino, M. Camurri, V. Barasuol, C. Mastalli, D. G. Caldwell, and C. Semini, "Heuristic Planning for Rough Terrain Locomotion in Presence of External Disturbances and Variable Perception Quality," 2018.

[3]https://youtu.be/daiarabCCx0