Feasibility-Aware Plan Adaptation in Humanoid Gait Generation

Michele Cipriano, Marcos R. O. A. Maximo, Nicola Scianca, Leonardo Lanari, Giuseppe Oriolo

2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids) | 979-8-3503-0327-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/HUMANOIDS57100.2023.10375146

***Abstract*— Most available schemes used for humanoid walking rely on the separation into a planning phase, typically offline, and a Model Predictive Controller (MPC). Moreover, in order for the MPC to work in real time, simplifying assumptions are made both on the template model and on the constraints so that the underlying optimization problem is a Quadratic Programming (QP). The planner is unaware of the underlying humanoid dynamics and of any disturbance acting on the robot. We present an on-line Feasibility-Aware Plan Adaptation (FAPA) module which can locally adapt footsteps (positions, timings and orientation) in such a way that it guarantees feasibility of the subsequent Intrinsically Stable MPC (IS-MPC) stage. We present two versions of the proposed scheme: one with a fixed regions assignment for placing the footstep and another one where the regions are selected automatically through mixed-integer programming. Simulation results show the effectiveness of the FAPA scheme.**

I. Introduction

Humanoid robot locomotion is a complex task that involves multiple concurrent activities. It is usually tackled by breaking it down into several subproblems and solving each of them more or less independently. The first component is in general a footstep planner, which determines a sequence of footstep, e.g., leading the robot to some desired location. This sequence of footsteps must be kinematically realizable at least in terms of step lengths. The humanoid dynamics are usually accounted for in a second stage, typically based on Model Predictive Control (MPC), using a simplified robot model which is used to generate Center of Mass (CoM) trajectories. MPC, in its basic form, allows to perform real-time footstep position adaptation [1] and obtain reactive stepping so to reject pushes and impacts. However, in order to be able to formulate the optimization problem as a Quadratic Program (QP), constraints should be kept linear. For this reason, most schemes only adapt footstep positions, leaving out footstep orientation and step timing.

Several efforts to improve this basic paradigm have been made. To include automatic step timing adaptation, one could make the MPC nonlinear [2], [3], [4], [5], denying real-time implementation or requiring significant compromise in the control rate. A linear formulation is obtainable by considering only the duration of the first footstep [6], [7]. As for footstep orientation, this is also often ignored or planned independently of the dynamics [1]. To couple rotation decision with the dynamics, some schemes employ non-convex optimization through nonlinear [8], [9] or Mixed-Integer Programming (MIP) [2]. MIP can also be used to alternatively select between multiple convex regions in which to place the footsteps, which would otherwise constitute a non-convex constraint [10], [11].

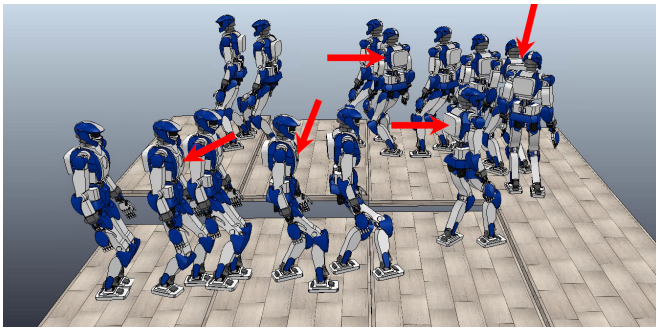

Fig. 1. An example simulation using the proposed architecture: the robot is walking along a staircase while being subject to multiple pushes. The adaptation module modifies position, orientation and timing of the footsteps real-time to guarantee a successful execution.

Our architecture is based on the Intrinsically Stable MPC (IS-MPC) of [12], which involves an explicit stability constraint ensuring the boundedness of the CoM trajectory with respect to the ZMP and is recursive feasibile. The feasibility region, i.e., the state space region for which the constrained QP admits a solution, can be used to enhance the scheme capabilities by adapting the timing of the first step [6], or to allow for non-convex regions [13] without burdening the optimization problem of the MPC.

In this paper, we add an online adaptation module that can locally adapt footsteps (positions, timings and orientation) so to guarantee feasibility of the subsequent IS-MPC stage. The Feasibility-Aware Plan Adaptation (FAPA) is thus dependent on the system state and the dynamics of the chosen template dynamic model.

We obtain the generality given by nonlinear constraints without sacrificing much performance as the number of variables in the planner is much lower than that of the variables of the MPC, making it very fast and capable of working in real time. Furthermore, we explore the inclusion of integer variables, further increasing the range of situations that can be covered.

Modules for online footstep adaptation using nonlinear optimization have been proposed [14], but not in conjunction with MPC. Our approach is not only designed to work along

Michele Cipriano, Nicola Scianca, Leonardo Lanari and Giuseppe Oriolo are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Italy. E-mail: *lastname*@diag.uniroma1.it.

Marcos Maximo is with the Autonomous Computational Systems Lab (LAB-SCA), Computer Science Division, Aeronautics Institute of Technology, Brazil. E-mail: *mmaximo*@ita.br.

Nicola Scianca has been fully supported by PNRR MUR project PE0000013-FAIR.

Marcos Maximo has been partially supported by CAPES and CNPq through grants 88887.717777/2022-00 and 307525/2022-8, respectively.

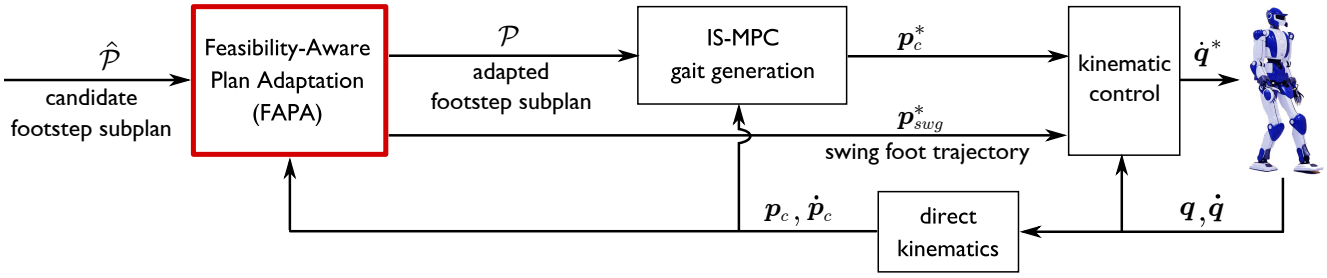

Fig. 2. A block scheme of the proposed architecture. The candidate footstep subplan $\hat{\mathcal{P}}$ is adapted by the FAPA module, guaranteeing the feasibility of IS-MPC. The IS-MPC module receives the adapted footstep subplan $\mathcal{P}$, and generates a desired trajectory of the CoM $\boldsymbol{p}_c^*$, which is used by the kinematic controller, together with the desired trajectory of the swing foot $\boldsymbol{p}_{\text{swg}}^*$, to generate the desired joint velocities $\dot{\boldsymbol{q}}^*$.

with the MPC module, but it specifically aimed at enhancing its capabilities.

The remainder of the paper is organized as follows. Sect. II gives a general formulation of the problem. Sect. III introduces some preliminary notions. Sect. IV details the MPC formulation. Sect. V describes both versions of the proposed FAPA: without and with integer variables. Sect. VI shows and discusses some simulations. Finally, Sect. VII presents concluding remarks and future extensions.

II. Problem Formulation

The proposed architecture is shown in Fig. 2. An external candidate plan is provided, which in this paper will be either a basic plan to demonstrate simple motions, or a plan generated by randomized exploration [15] for more complex environments. A subplan, i.e., a portion of the candidate plan, is given as input to the scheme at each timestep.

The basic components of the considered scheme are:

- a Feasibility-Aware Plan Adaptation (FAPA) block, that can modify locally the high-level footstep plan;
- an IS-MPC gait generation block that generates CoM/ZMP trajectories based on the output of FAPA;
- a kinematic controller that realizes at the joint level the generated CoM and swing foot trajectories.

While the high-level footstep plan is designed considering the humanoid's kinematic limitations, it is entirely unaware of its dynamics and is not informed by the robot state since it is fully generated off-line. To make up for this deficiency, the FAPA module performs a local adaptation of the planned footsteps before these enter the IS-MPC stage.

This adaptation is based on a *gait feasibility constraint* that guarantees feasibility of the next IS-MPC stage while trying to match the original plan. It can concurrently change the footstep positions, orientations, as well as step timings.

To formulate this constraint, we leverage the feasibility region of IS-MPC, i.e., the subset of the state space where the problem is feasible at a given time. While in previous analyses we provided approximate closed-form expressions for these region bounds, here we rather define the feasibility region in an implicit form with the nonlinear dependency on the footstep positions, orientations, and step timings.

The fact that the feasibility of the MPC can be efficiently captured by the expression of this constraint is a crucial aspect of the formulation, because it means that the scheme can harness the power of nonlinear optimization without burdening the MPC itself, which remains linear and can run at a high rate. The nonlinear optimization part is external to the MPC, which allows the number of variables to be kept small and thus to keep the computation time manageable.

We propose two versions of the FAPA module, that differ by the optimization problem required for their implementation. In particular, the first version only uses continuous optimization, while the second one also employs discrete variables and is formulated as a Mixed Integer Nonlinear Program (MINLP). Being the latter very general it can be used to account for more adaptation scenarios, e.g., in which the footsteps can also be moved to different terrain patches than the ones assigned by the high-level planner. As will be discussed extensively in Sect. VI, the second version is more demanding in terms of computation time, but we present it as a proof of concept as we strongly believe it can be made to work in real time with proper code optimization.

III. Preliminaries

In this section we describe the environment and the structure of the footstep plan used in our scheme.

A. Environment

The considered environment is a world of stairs, i.e., constituted by flat horizontal regions. The robot is allowed to walk across different regions if these are relatively close in height, and if there is sufficient available surface to step on them, otherwise they will constitute obstacles to be avoided.

The arrangement of these regions is assumed to be known, and it is processed and encoded in the following way:

- regions are reduced in size so that they represent the collision-free area available for the center of the footprint. This is done by performing a Minkowski difference between each flat region and the area swept by a footprint (accounting for all possible footstep orientations);
- after reduction, non-convex regions are subdivided into non-overlapping convex polytopic *patches*.

A patch P is identified by the inequality $\boldsymbol{A}(P)\boldsymbol{p} \leq \boldsymbol{b}(P)$, where $\boldsymbol{A}(P) \in \mathbb{R}^{V(P)\times 2}$ and $\boldsymbol{b}(P) \in \mathbb{R}^{V(P)}$ define a polytope (with $V(P)$ vertices) and $\boldsymbol{p} = (x, y)^T$ is a generic 2D point. In this way, non-polytopic portions of ground (e.g., round edges) are approximated, but the number of vertices can be arbitrarily large. Since each patch P is flat, its height is denoted simply as $z(P)$.

B. Footstep plan

The high-level footstep plan is a sequence of *candidate footsteps* $\hat{\boldsymbol{f}}$, each identified by the tuple $\hat{\boldsymbol{f}} = (\hat{x}_f, \hat{y}_f, \hat{z}_f, \hat{\theta}_f, \hat{T}_{\rm ss}, \hat{T}_{\rm ds})$. For each planned footstep $\hat{\boldsymbol{f}}$

- $\hat{x}_f$, $\hat{y}_f$ and $\hat{z}_f$ are the coordinates of its center;
- $\hat{\theta}_f$ is its orientation around the z axis;
- $\hat{T}_{\rm ds}$ and $\hat{T}_{\rm ss}$ are the durations of its double support and single support phases, respectively;
- we denote by $\Pi(\hat{\boldsymbol{f}})$ the patch that contains the footstep, i.e., the patch P such that[1]

$$(\hat{x}_f, \hat{y}_f)^T \in P, \quad \hat{z}_f = z(P).$$

The footstep plan $\hat{\mathcal{P}}$ is computed off-line, and at each time t_k a subplan $\hat{\mathcal{P}}^l$ of size $F+1$ is extracted, where l is the index of the first footstep of the current subplan (at t_k), and F a fixed parameter. The subplan contains the next F candidate footsteps:

$$\hat{\mathcal{P}}^l = \left\{ \hat{\boldsymbol{f}}^l, \ldots, \hat{\boldsymbol{f}}^{l+F} \right\}.$$

The FAPA block, which performs footsteps adaptation, modifies $\hat{\mathcal{P}}^l$ in the adapted subplan $\mathcal{P}^l$, i.e., in the input of the IS-MPC block

$$\mathcal{P}^l = \left\{ \boldsymbol{f}^l, \ldots, \boldsymbol{f}^{l+F} \right\}.$$

After every iteration, if adaptation took place (i.e., $\mathcal{P}^l$ differs from $\hat{\mathcal{P}}^l$), the algorithm performs a *footstep plan override*, i.e., the corresponding portion of the high-level footstep plan is substituted with the adapted subplan $\mathcal{P}^l$. Note that the remaining part of the plan (after the index $l+F$) is unchanged, so if the adaptation makes the robot stray from the initial path it will later try to catch up. This behavior is often acceptable, but might sometimes be undesirable, and can be improved in future versions if we allow the high-level planner to replan on-line (see [15]).

IV. Gait Generation via IS-MPC

We first illustrate the IS-MPC block in order to introduce concepts that are necessary to explain the FAPA block.

We describe the prediction model, the constraints, and the optimization problem to be solved. Furthermore, we give an expression for the feasibility region of IS-MPC in a suitable form to be used by the proposed scheme.

A. Prediction Model

The prediction model is derived from balancing moments around the ZMP. To allow vertical motion of the CoM, some works [16] use the VH-IP model, in which height variations cause a change to the natural frequency of an inverted pendulum. This makes the prediction model nonlinear, negatively impacting performance. It is however possible to generate 3D trajectories using a linear model if we constrain the systems to obey the linear dynamics

$$\ddot{\boldsymbol{p}}_c = \eta^2 (\boldsymbol{p}_c - \boldsymbol{p}_z) + \boldsymbol{g}, \tag{1}$$

[1]Note that this patch is unique because the environment is subdivided into non-overlapping patches.

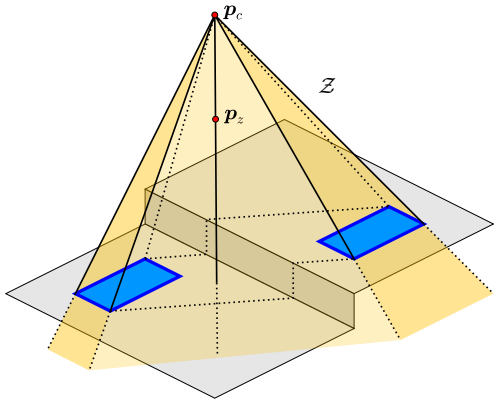

Fig. 3. 3D balance: the ZMP $\boldsymbol{p}_z$ must be inside the pyramid $\mathcal{Z}$.

as in [15], where $\boldsymbol{p}_c = (x_c, y_c, z_c)^T$ is the CoM position, $\boldsymbol{p}_z = (x_z, y_z, z_z)^T$ is the ZMP position, η is a design parameter[2] and $\boldsymbol{g} = (0, 0, -9.81)^T$ [m/s^2] is the gravity acceleration vector. In order to have smoother trajectories, we dynamically extend (1) by having the derivative of the ZMP $\dot{\boldsymbol{p}}_z$, instead of the ZMP itself, be the input of the model.

Through the change of coordinates $\boldsymbol{p}_u = \boldsymbol{p}_c + \dot{\boldsymbol{p}}_c/\eta$, the dynamics of the unstable component (also *divergent component of motion* [17] or *capture point* [18]) can be highlighted:

$$\dot{\boldsymbol{p}}_u = \eta(\boldsymbol{p}_u - \boldsymbol{p}_z) + \boldsymbol{g}/\eta.$$

Despite this unstable dynamics, the CoM trajectory is bounded with respect to the ZMP if the following *stability condition* is satisfied:

$$\boldsymbol{p}_u^k = \eta \int_{t_k}^{\infty} e^{-\eta(\tau - t_k)} \boldsymbol{p}_z(\tau) d\tau - \frac{\boldsymbol{g}}{\eta}, \tag{2}$$

where the superscript in $\boldsymbol{p}_u^k$ indicates that the variable is sampled at time t_k.

The MPC module works over discrete time-steps of duration δ, over which the input $\dot{\boldsymbol{p}}_z$ is constant, i.e., $\dot{\boldsymbol{p}}_z(t) = \dot{\boldsymbol{p}}_z^k$ for $t \in [t_k, t_{k+1})$. The prediction model (1) is used to forecast the evolution of the system over a *control horizon* $T_c = C\delta$, while we assume to know the footstep plan over a *preview horizon* $T_p = P\delta$, with $P \geq C$.

Relating the dynamics of the CoM to those of the ZMP is essential since the latter encodes information about the realizability of ground reaction forces, and thus provides a criterion for balance. A common way to extend the basic 2D balance criterion consists in prescribing the 3D ZMP to be inside a 3D pyramid $\mathcal{Z}$ (see Fig. 3), having the base defined by the contact surfaces and the CoM vertex [19], [15].

B. ZMP Constraint

The IS-MPC block receives the adapted subplan $\mathcal{P}^l$ and uses it to construct ZMP constraints. As described in the previous subsection, the criterion for balance is satisfied if the ZMP belongs to the pyramid $\mathcal{Z}$. However, enforcing this condition directly would lead to a nonlinear constraint in the

[2]The parameter η should be chosen by taking into account the desired resting height of the humanoid, because eq. (1) has an equilibrium when CoM and ZMP are vertically displaced by $\boldsymbol{g}/\eta^2$.

MPC because the vertex of the pyramid is the CoM of the robot. Thus, we adopt a conservative approximation called the *moving constraint*.

The moving constraint requires for the ZMP to be at all times within a convex polyhedron of fixed shape, in our case a box of dimensions d_x, d_y and d_z centered in $\boldsymbol{p}_{\rm mc} = (x_{\rm mc}, y_{\rm mc}, z_{\rm mc})$, which we call the *moving box*. Along the prediction, the moving box can translate but not rotate, and its center moves in such a way that it is always fully contained within the 3D pyramid $\mathcal{Z}$ (see Fig. 5 in [20]). The vector $\boldsymbol{X}_{\rm mc}^{k+1} = (x_{\rm mc}^{k+1}, \dots, x_{\rm mc}^{k+C})^T$ collects the x coordinate of the center of the moving box in the control horizon.

Because of its constant orientation in the prediction, at each time we can choose the orientation of the axes to align with the orientation of the moving box (taken as the orientation of the current support foot) and obtain a ZMP constraint that is decoupled along the 3 axes. Focusing on the component along x, we can write it as

$$\boldsymbol{X}_z^{\rm m,k+1} \leq \boldsymbol{X}_z^{k+1} \leq \boldsymbol{X}_z^{M,k+1}, \tag{3}$$

where $\boldsymbol{X}_z^{k+1} = (x_z^{k+1}, \dots, x_z^{k+C})^T$ is a vector of predicted ZMP positions, and $\boldsymbol{X}_z^{\rm m,k+1}$ and $\boldsymbol{X}_z^{\rm M,k+1}$ are the ZMP bounds along the prediction. By defining

$$\boldsymbol{Z} = \begin{pmatrix} \delta & 0 & \cdots & 0 \\ \delta & \delta & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \delta & \delta & \cdots & \delta \end{pmatrix}, \qquad \boldsymbol{z} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix},$$

$\boldsymbol{X}_z^{k+1}$ can be expressed as

$$\boldsymbol{X}_z^{k+1} = \boldsymbol{Z}\dot{\boldsymbol{X}}_z^k + \boldsymbol{z}x_z^k, \tag{4}$$

where $\dot{\boldsymbol{X}}_z^k = (\dot{x}_z^k, \dots, \dot{x}_z^{k+C-1})^T$ is the vector of ZMP velocities, i.e., the decision variables. The ZMP bounds along the prediction can be expressed as

$$\boldsymbol{X}_z^{\rm m,k+1} = \boldsymbol{X}_{\rm mc}^{k+1} - \boldsymbol{z}\frac{d_x}{2}, \quad \boldsymbol{X}_z^{\rm M,k+1} = \boldsymbol{X}_{\rm mc}^{k+1} + \boldsymbol{z}\frac{d_x}{2}. \tag{5}$$

The center of the moving box $\boldsymbol{p}_{\rm mc}$ must be expressed in terms of the subplan $\mathcal{P}^l$. First we define the *piecewise-linear sigmoid* function

$$\sigma(t, t_i, t_f) = \frac{1}{t_f - t_i}\left(\rho(t - t_i) - \rho(t - t_f)\right),$$

where $\rho(t) = t\delta_{-1}(t)$ is the unit ramp. $\sigma(t, t_i, t_f)$ is 0 before t_i, 1 after t_f, and it transitions linearly in the interval $[t_i, t_f]$. This function is useful to represent the transition between consecutive footsteps.

$\boldsymbol{X}_{\rm mc}^{k+1}$ can be written as

$$\boldsymbol{X}_{\rm mc}^{k+1} = \boldsymbol{M}\boldsymbol{X}_f^l + \boldsymbol{m}x_f^l, \tag{6}$$

where $\boldsymbol{X}_f^l = (x_f^l, \dots, x_f^{l+F})^T$ collects the footstep positions. $\boldsymbol{M} \in \mathbb{R}^{C\times F}$ is a mapping matrix whose elements M_{ij} are defined as

$$\begin{aligned} M_{ij} = {} & \sigma(t_{k+i}, t_s^{l+j}, t_s^{l+j} + T_{\rm ds}^{l+j}) \\ & - \sigma(t_{k+i}, t_s^{l+j-1}, t_s^{l+j-1} + T_{\rm ds}^{l+j-1}), \end{aligned} \tag{7}$$

and $\boldsymbol{m} \in \mathbb{R}^C$ is a vector whose elements m_i are given by

$$m_i = 1 - \sigma(t_{k+i}, t_s^l, t_s^l + T_{\rm ds}^1),$$

where t_s^l is the starting time of the l-th step and

$$t_s^j = t_s^l + \sum_{\lambda=l}^{l+j-1}\left(T_{\rm ds}^\lambda + T_{\rm ss}^\lambda\right).$$

C. Stability Constraint

The stability constraint is derived from the stability condition (2). We focus here on the x component, but the other components can be similarly derived (see [15]).

Since the MPC has a limited control horizon we split the integral at t_{k+C}. Before t_{k+C} the stability condition can be expressed in terms of the decision variables, while after it must be conjectured. We adopt the anticipative tail $\tilde{x}_z$ [12] to get an approximation of the ZMP trajectory after the horizon:

$$\eta\int_{t_k}^{t_{k+C}} e^{-\eta(\tau - t_k)}x_z(\tau)d\tau = x_u^k - \tilde{c}_x^k,$$

where x_u^k is the x-component of $\boldsymbol{p}_u^k$ and

$$\tilde{c}_x^k = \eta\int_{t_{k+C}}^{\infty} e^{-\eta(\tau - t_k)}\tilde{x}_z(\tau)d\tau.$$

The second step is to make the decision variables appear explicitly, i.e., the ZMP velocities $\dot{\boldsymbol{X}}_z$ over the control horizon, by computing the integral over a piecewise linear ZMP trajectory. The final form of the constraint can be found in [12]. For the purpose of this analysis, we will use the compact expression

$$\boldsymbol{s}^T\dot{\boldsymbol{X}}_z = b_x^k + x_u^k, \tag{8}$$

where $\boldsymbol{s} \in \mathbb{R}^C$ and $b_x^k \in \mathbb{R}$ denote respectively a vector and a scalar whose explicit expressions can be recovered from the cited reference.

D. IS-MPC Algorithm

IS-MPC solves, at each time t_k, the following QP problem:

$$\begin{cases} \min\limits_{\dot{\boldsymbol{X}}_z^k, \dot{\boldsymbol{Y}}_z^k, \dot{\boldsymbol{Z}}_z^k} \|\dot{\boldsymbol{X}}_z^k\|^2 + \|\dot{\boldsymbol{Y}}_z^k\|^2 + \|\dot{\boldsymbol{Z}}_z^k\|^2 + \beta\|\boldsymbol{X}_z^{k+1} - \boldsymbol{X}_{\rm mc}^{k+1}\|^2 \\ \qquad + \beta\|\boldsymbol{Y}_z^{k+1} - \boldsymbol{Y}_{\rm mc}^{k+1}\|^2 + \beta\|\boldsymbol{Z}_z^{k+1} - \boldsymbol{Z}_{\rm mc}^{k+1}\|^2 \\ \text{subject to:} \\ \bullet\ \text{ZMP constraints (3)} \\ \bullet\ \text{stability constraints (8)} \end{cases}$$

In the cost function, the first three terms act as regularization while the remaining attempt to bring the ZMP as close as possible to the center of the moving box, with a strength modulated by the weight β.

The first sample $\dot{\boldsymbol{p}}_z^k = (\dot{x}_z^k, \dot{y}_z^k, \dot{z}_z^k)$ of the optimal sequence is used to integrate the prediction model and the resulting CoM position $\boldsymbol{p}_c^{k+1}$ is sent to the kinematic controller together with a suitable swing foot trajectory that allows to reach the target footstep position at the proper time.

E. Feasibility Region

The *feasibility region* is the region of the state space in which the IS-MPC optimization problem is feasible.

Proposition 1: IS-MPC is feasible at time t_k if

$$\begin{aligned}
&\boldsymbol{s}^T\boldsymbol{Z}^{-1}(\boldsymbol{X}_z^{\mathrm{m},k+1}-\boldsymbol{z}x_z^k)\le x_u^k+b_x^k\le \boldsymbol{s}^T\boldsymbol{Z}^{-1}(\boldsymbol{X}_z^{\mathrm{M},k+1}-\boldsymbol{z}x_z^k),\\
&\boldsymbol{s}^T\boldsymbol{Z}^{-1}(\boldsymbol{Y}_z^{\mathrm{m},k+1}-\boldsymbol{z}y_z^k)\le y_u^k+b_y^k\le \boldsymbol{s}^T\boldsymbol{Z}^{-1}(\boldsymbol{Y}_z^{\mathrm{M},k+1}-\boldsymbol{z}y_z^k),\\
&\boldsymbol{s}^T\boldsymbol{Z}^{-1}(\boldsymbol{Z}_z^{\mathrm{m},k+1}-\boldsymbol{z}z_z^k)\le z_u^k+b_z^k\le \boldsymbol{s}^T\boldsymbol{Z}^{-1}(\boldsymbol{Z}_z^{\mathrm{M},k+1}-\boldsymbol{z}z_z^k).
\end{aligned} \tag{9}$$

Proof. We focus the proof on the inequalities for the x component, as the logic, for the other components is identical. The bounds of the feasibility region along x are given by

$$\begin{aligned}
x_u^{k,b1} &= \boldsymbol{s}^T\boldsymbol{Z}^{-1}(\boldsymbol{X}_z^{\mathrm{m},k+1}-\boldsymbol{z}x_z^k)-b_x^k,\\
x_u^{k,b2} &= \boldsymbol{s}^T\boldsymbol{Z}^{-1}(\boldsymbol{X}_z^{\mathrm{M},k+1}-\boldsymbol{z}x_z^k)-b_x^k.
\end{aligned}$$

Then, if x_u^k is inside the feasibility region, it is possible to express it as a convex combination of the two bounds, i.e.,

$$x_u^k = \alpha x_u^{k,b1}+(1-\alpha)x_u^{k,b2}, \alpha\in[0,1]. \tag{10}$$

Consider the following ZMP velocity trajectory:

$$\dot{\boldsymbol{X}}_z^k = \alpha\boldsymbol{Z}^{-1}(\boldsymbol{X}_z^{\mathrm{m},k+1}-\boldsymbol{z}x_z^k)+(1-\alpha)\boldsymbol{Z}^{-1}(\boldsymbol{X}_z^{\mathrm{M},k+1}-\boldsymbol{z}x_z^k). \tag{11}$$

We will show that this particular trajectory satisfies both the stability constraint and the ZMP constraints. As for the stability constraint, multiply both sides of (11) by $\boldsymbol{s}^T$ and plug in the definitions of $x_u^{k,b1}$ and $x_u^{k,b2}$ to obtain

$$\boldsymbol{s}^T\dot{\boldsymbol{X}}_z^k = (\alpha(x_u^{k,b1}+b_x^k))+(1-\alpha)(x_u^{k,b2}+b_x^k)).$$

Using (10), this is equivalent to the stability constraint (8).

To prove satisfaction of the ZMP constraint. Left-multiplying (4) by $\boldsymbol{Z}$, the chosen ZMP velocity trajectory can be rewritten as

$$\boldsymbol{X}_z^k-\boldsymbol{z}x_z^k = \alpha(\boldsymbol{X}_z^{\mathrm{m},k+1}-\boldsymbol{z}x_z^k)+(1-\alpha)(\boldsymbol{X}_z^{\mathrm{M},k+1}-\boldsymbol{z}x_z^k),$$

which simplifies to $\boldsymbol{X}_z^k = \alpha\boldsymbol{X}_z^{\mathrm{m},k+1}+(1-\alpha)\boldsymbol{X}_z^{\mathrm{M},k+1}$, and therefore the ZMP constraint (3) is satisfied. ∎

In the following section, we will describe how to use the feasibility region to formulate a constraint for the FAPA module, and thus ensure that the output of FAPA can be used by IS-MPC to construct a feasible QP.

V. Feasibility-Aware Plan Adaptation

The FAPA module runs in real-time and performs a local adaptation of the subplan $\hat{\mathcal{P}}^l$, including their timing. We now describe the constraints and the optimization problems that define the adaptation procedure.

A. Kinematic Constraint

The j-th footstep $\boldsymbol{f}^j$ is ensured to be kinematically feasible by limiting its displacement with respect to the previous footstep $\boldsymbol{f}^{j-1}$. In practice we constrain the geometric components of $\boldsymbol{f}^j$ to be within the *admissible region*

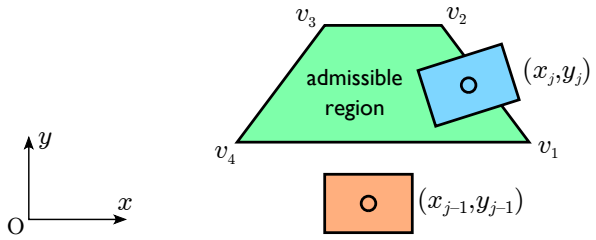

Fig. 4. Admissible region of the kinematic constraint in the x-y plane.

$$\begin{aligned}
&\begin{pmatrix}\boldsymbol{n}_1^T\left(\boldsymbol{p}_{xy}^{l+j}-\boldsymbol{p}_{xy}^{l+j-1}-\boldsymbol{R}(\theta_f^{l+j-1})\boldsymbol{v}_1\right)\\ \vdots\\ \boldsymbol{n}_V^T\left(\boldsymbol{p}_{xy}^{l+j}-\boldsymbol{p}_{xy}^{l+j-1}-\boldsymbol{R}(\theta_f^{l+j-1})\boldsymbol{v}_V\right)\end{pmatrix}\ge\boldsymbol{0},\\
&\Delta z^{\mathrm{m}}\le z_{l+j}-z_{l+j-1}\le\Delta z^{\mathrm{M}},\\
&\Delta\theta^{\mathrm{m}}\le\theta_{l+j}-\theta_{l+j-1}\le\Delta\theta^{\mathrm{M}},
\end{aligned} \tag{12}$$

with $\boldsymbol{R}(\theta_f^{l+j-1})$ a 2D rotation matrix, $\boldsymbol{n}_i$ the vector normal to the i-th segment of the convex region computed as

$$\boldsymbol{n}_i = \begin{pmatrix}0 & -1\\ 1 & 0\end{pmatrix}\boldsymbol{R}(\theta_f^{l+j-1})(\boldsymbol{v}_{i+1}-\boldsymbol{v}_i),$$

and $\boldsymbol{v}_i$ being the vertices defining the convex polygon (different depending whether the support foot is left or right), shown in Fig. 4. Furthermore, Δ_z^m, Δz^{M}, $\Delta\theta^{\mathrm{m}}$, and $\Delta\theta^{\mathrm{M}}$ define limits for the foot reachability over vertical displacement and relative orientation.

B. Timing Constraint

Single and double support duration are subject to minimum and maximum duration constraints

$$T_{\mathrm{ds}}^{\min}\le T_{\mathrm{ss}}^{l+j}\le T_{\mathrm{ds}}^{\max},\quad T_{\mathrm{ss}}^{\min}\le T_{\mathrm{ds}}^{l+j}\le T_{\mathrm{ss}}^{\max}, \tag{13}$$

where the bounds $T_{\mathrm{ds}}^{\min}$, $T_{\mathrm{ds}}^{\max}$, $T_{\mathrm{ss}}^{\min}$, $T_{\mathrm{ss}}^{\max}$ are chosen in such a way to avoid excessively fast trajectories that might be difficult to track, as well as very slow steps that could result in quasi-static motion.

C. Patch Constraints

We describe alternative versions of this constraint, as we will later compare the module using either of them, both in terms of the quality of the resulting plan and of the computational load. The first version of the constraint simply restrict the $(l+j)$-th footstep to lie within its associated patch $\Pi(\boldsymbol{f}^{l+j})$, which is the one originally chosen by the high-level planner. This constraint can be written as

$$\begin{cases}\boldsymbol{A}(\Pi(\boldsymbol{f}^{l+j}))\begin{pmatrix}x_f^{l+j} & y_f^{l+j}\end{pmatrix}^T\le\boldsymbol{b}(\Pi(\boldsymbol{f}^{l+j})),\\ z_f^{l+j}=z(\Pi(\boldsymbol{f}^{l+j})).\end{cases} \tag{14}$$

The second version of the patch constraint allows the footstep to be moved to a different patch. To entertain this possibility, we introduce binary variables in order to formulate a mixed-integer constraint. This constraint defines

a logical implication in which, if a certain binary variable $b_{l+j,\kappa}$ is *true*, then a linear constraint must be verified:

$$b_{l+j,\kappa} = 1 \Rightarrow \begin{cases} \boldsymbol{A}(P^\kappa)\begin{pmatrix} x_f^{l+j} & y_f^{l+j} \end{pmatrix}^T \leq \boldsymbol{b}(P^\kappa), \\ z_f^{l+j} = z(P^\kappa). \end{cases} \tag{15}$$

This forces the $(l+j)$-th footstep to lie within the κ-th patch. Since each footstep can only be inside a single patch, we also impose

$$\sum_{\kappa=1}^{R} b_{l+j,\kappa} = 1. \tag{16}$$

In MIP, logical implications can be implemented using binary variables through the so-called *big-M* technique [21]. In this case, we rewrite (15) as

$$\begin{cases} \boldsymbol{A}(P^\kappa)\begin{pmatrix} x_f^{l+j} & y_f^{l+j} \end{pmatrix}^T \leq \boldsymbol{b}(P^\kappa) + (1 - b_{l+j,\kappa})M\mathbf{1}_{V(P^\kappa)}, \\ z_{l+j} \leq z(P^\kappa) + (1 - b_{l+j,\kappa})M, \\ -z_{l+j} \leq -z(P^\kappa) + (1 - b_{l+j,\kappa})M, \end{cases} \tag{17}$$

where M is a constant large enough to relax the constraints if $b_{l+j,\kappa} = 0$ and $\mathbf{1}_{V(P^\kappa)}$ is a row vector with $V(P^\kappa)$ ones. We define $\hat{\kappa}_{l+j}$ as the index of $\Pi(\hat{\boldsymbol{f}}^{l+j})$. Note that this requires turning the equality constraint into two inequality constraints. Based on the patches of the candidate footsteps in $\hat{\mathcal{P}}^l$, we also define candidate binary variables as

$$\hat{b}_{l+j,\kappa} = \begin{cases} 1, \text{ if } \kappa = \hat{\kappa}_{l+j}, \\ 0, \text{ if } \kappa \neq \hat{\kappa}_{l+j}. \end{cases}$$

Finally, (16) and (17) assume that every footstep may be mapped to every patch, which requires $F \times R$ binary variables. However, since the computational load of a MIP is largely related to the number of binary variables, we employ a heuristic that allows a footstep $\boldsymbol{f}^j$ to be assigned only to the patches adjacent to $\Pi(\hat{\boldsymbol{f}}^j)$.

D. Current Footstep Constraints

The first footstep in the subplan $\boldsymbol{f}^l$ corresponds to the footstep currently in contact with the ground, which means that some of its components cannot be changed. In particular, its geometric components should be constrained to be equal to the corresponding components of $\hat{\boldsymbol{f}}^l$, i.e.,

$$x_f^l = \hat{x}_f^l, \quad y_f^l = \hat{y}_f^l, \quad z_f^l = \hat{z}_f^l, \quad \theta_f^l = \hat{\theta}_f^l. \tag{18}$$

Note that, because of the footstep plan override, the components of $\hat{\boldsymbol{f}}^l$ are not the same as in the original plan, but rather those adapted at the previous iteration.

If t_k belongs to a single support phase, the double support of the current step cannot be changed anymore because it is already passed. This is expressed by the constraint

$$t_k - t_s^l > T_{\text{ds}}^l \quad \Rightarrow \quad T_{\text{ds}}^l = \hat{T}_{\text{ds}}^l. \tag{19}$$

Note that the implication in (19) is handled at the code level and does not require introducing binary variables.

To avoid footstep changes when the swing foot is close to touching the ground, when nearing the end we add the following constraint:

$$T_{\text{ds}}^l + T_{\text{ss}}^l - t_k + t_s^l < t_{\text{change}} \quad \Rightarrow \quad \boldsymbol{f}^{l+1} = \hat{\boldsymbol{f}}^{l+1}. \tag{20}$$

E. Gait Feasibility Constraints

The gait feasibility constraints are introduced to ensure that IS-MPC is feasible. They do so by constraining the current state to be within the feasibility region (9).

The expression of the feasibility region (9) uses the ZMP bounds, that clearly depend on the motion of the moving box, and thus on the footsteps positions and timings. To derive a constraint, we simply make this dependency explicit by plugging (5) and (6) inside (9). Focusing on the right inequality of the x component, this results in

$$x_u^k + b_x^k \leq \boldsymbol{s}^T \boldsymbol{Z}^{-1}\left(\boldsymbol{M}\boldsymbol{X}_f^l + \boldsymbol{m}x_f^l + \boldsymbol{z}\left(\frac{d_x}{2} - x_z^k\right)\right). \tag{21}$$

The y and z components, as well as the left inequalities result in analogous expressions, which we omit for space concerns.

F. Feasibility-Driven Plan Adaptation Algorithm

We present two different versions of the FAPA algorithm. The first one is not allowed to move footsteps from a different patch to the one in the original plan, and is thus referred to as Fixed patches FAPA (F-FAPA). The second one is instead allowed to choose different patches, and goes under the name of Variables patches FAPA (V-FAPA).

The decision variable over the planning horizon are collected as

$$\begin{aligned} \boldsymbol{X}_f^l &= (x_f^l, \ldots, x_f^{l+F}), \quad \boldsymbol{Y}_f^l = (y_f^l, \ldots, y_f^{l+F}), \\ \boldsymbol{Z}_f^l &= (z_f^l, \ldots, z_f^{l+F}), \quad \boldsymbol{\Theta}_f^l = (\theta_f^l, \ldots, \theta_f^{l+F}), \\ \boldsymbol{T}_{\text{ds}}^l &= (T_{\text{ds}}^l, \ldots, T_{\text{ds}}^{l+F}), \quad \boldsymbol{T}_{\text{ss}}^l = (T_{\text{ss}}^l, \ldots, T_{\text{ss}}^{l+F}), \end{aligned}$$

$$\boldsymbol{B}^l = \begin{pmatrix} b_{l,1} & \ldots & b_{l,L} \\ \vdots & \ddots & \vdots \\ b_{l+F,1} & \ldots & b_{l+F,L} \end{pmatrix},$$

while the corresponding candidate values are identified by the vectors $\hat{\boldsymbol{X}}_f^l$, $\hat{\boldsymbol{Y}}_f^l$, $\hat{\boldsymbol{Z}}_f^l$, $\hat{\boldsymbol{\Theta}}_f^l$, $\hat{\boldsymbol{T}}_{\text{ds}}^l$, $\hat{\boldsymbol{T}}_{\text{ss}}^l$, $\hat{\boldsymbol{B}}^l$, similarly defined.

F-FAPA solves the following problem, with decision variables $\boldsymbol{U}^l = (\boldsymbol{X}_f^l, \boldsymbol{Y}_f^l, \boldsymbol{Z}_f^l, \boldsymbol{\Theta}_f^l, \boldsymbol{T}_{\text{ds}}^l, \boldsymbol{T}_{\text{ss}}^l)$:

$$\begin{cases} \min\limits_{\boldsymbol{U}^l} \quad w_x\|\hat{\boldsymbol{X}}_f^l - \boldsymbol{X}_f^l\|^2 + w_y\|\hat{\boldsymbol{Y}}_f^l - \boldsymbol{Y}_f^l\|^2 + \\ \qquad\quad w_z\|\hat{\boldsymbol{Z}}_f^l - \boldsymbol{Z}_f^l\|^2 + w_\theta\|\hat{\boldsymbol{\Theta}}_f^l - \boldsymbol{\Theta}_f^l\|^2 + \\ \qquad\quad w_{\text{ds}}\|\hat{\boldsymbol{T}}_{\text{ds}}^l - \boldsymbol{T}_{\text{ds}}^l\|^2 + w_{\text{ss}}\|\hat{\boldsymbol{T}}_{\text{ss}}^l - \boldsymbol{T}_{\text{ss}}^l\|^2 \\ \text{subject to:} \\ \bullet \text{ kinematic constraints (12), for } j = 1, \ldots, F \\ \bullet \text{ timing constraints (13), for } j = 0, \ldots, F \\ \bullet \text{ fixed patch constraints (14), for } j = 1, \ldots, F \\ \bullet \text{ current footsteps constraints (18), (19) and (20)} \\ \bullet \text{ gait feasibility constraints (21)} \end{cases}$$

Since F-FAPA does not have binary variables, it can be implemented using a regular nonlinear solver (i.e., `ipopt`).

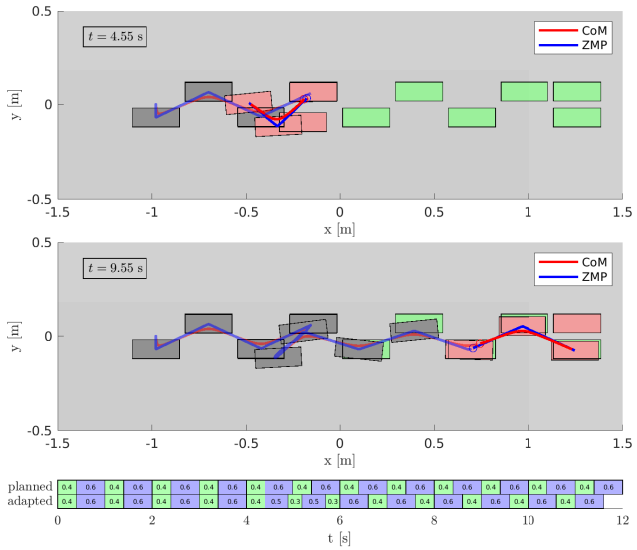

Fig. 5. F-FAPA in the *empty* scenario. The robot is walking in a straight line and is pushed at time 4.5 s (slightly before the first snapshot). Green footsteps represent the original candidate plan, while the footsteps that are actually executed are shown in grey. Red footsteps represent the current adapted subplan. The two bands on the bottom show the nominal and adapted timings (green for double support and blue for single support). The same color scheme is used for the rest of the figures.

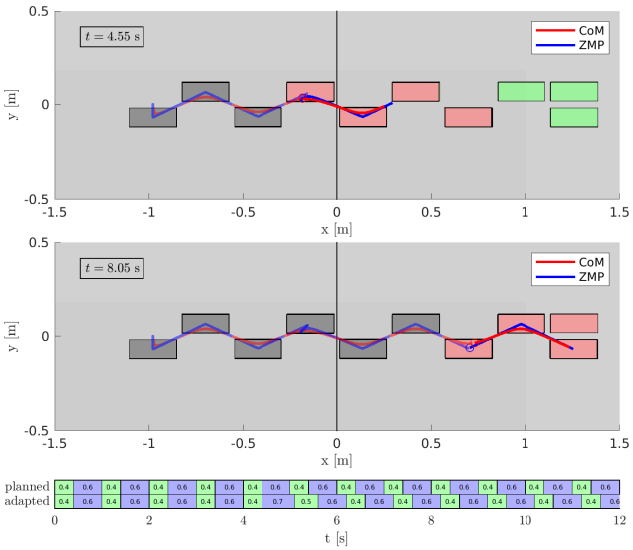

Fig. 6. F-FAPA in the *2-patches* scenario. The robot is walking in a straight line and is pushed at time 4.5 s (slightly before the first snapshot). Since changing patches is not allowed, the magnitude of the push that can be tolerated is quite small, compared to that of the other simulations.

V-FAPA solves the following problem, with decision variables which now include the binary variables $\boldsymbol{B}^l$ that is $\boldsymbol{W}^l = (\boldsymbol{X}_f^l, \boldsymbol{Y}_f^l, \boldsymbol{Z}_f^l, \boldsymbol{\Theta}_f^l, \boldsymbol{T}_{\text{ds}}^l, \boldsymbol{T}_{\text{ss}}^l, \boldsymbol{B}^l)$:

$$
\begin{aligned}
\min_{\boldsymbol{W}^l} \quad & w_x \|\hat{\boldsymbol{X}}_f^l - \boldsymbol{X}_f^l\|_2^2 + w_y \|\hat{\boldsymbol{Y}}_f^l - \boldsymbol{Y}_f^l\|_2^2 + \\
& w_z \|\hat{\boldsymbol{Z}}_f^l - \boldsymbol{Z}_f^l\|_2^2 + w_\theta \|\hat{\boldsymbol{\Theta}}_f^l - \boldsymbol{\Theta}_f^l\|_2^2 + \\
& w_{\text{ds}} \|\hat{\boldsymbol{T}}_{\text{ds}}^l - \boldsymbol{T}_{\text{ds}}^l\|_2^2 + w_{\text{ss}} \|\hat{\boldsymbol{T}}_{\text{ss}}^l - \boldsymbol{T}_{\text{ss}}^l\|_2^2 + \\
& w_{\text{b}} \|\hat{\boldsymbol{B}}^l - \boldsymbol{B}^l\|_2^2
\end{aligned}
$$

subject to:

- kinematic constraints (12), for $j = 1, \ldots, F$
- timing constraints (13), for $j = 0, \ldots, F$
- variable patch constraints (16) and (17), for $j = 1, \ldots, F$
- current footsteps constraints (18), (19) and (20)
- gait feasibility constraints (21)

Since V-FAPA contains the binary variables $\boldsymbol{B}$ it is implemented as a MINLP.

VI. Simulations

We ran four simulations in MATLAB, using CoppeliaSim to kinematically visualize the resulting motions. The system is an AMD Ryzen 9 5900X (4.8 GHz, 12 core) with 16 GB DDR4 3600 MHz running Ubuntu 22.04 LTS. IS-MPC runs at 100 Hz and is solved using `quadprog`, while FAPA runs at 10 Hz and is solved using the CasADi interface. In CasADi, we used `ipopt` for F-FAPA, and `bonmin` for V-FAPA. We also ran tests with the commercial solver `knitro`, to compare the performance (see Table I).

All the simulations use the following parameters: $\delta = 0.01$ s, $T_c = 2.0$ s, $T_p = 4.0$ s, $\eta = 3.6$ s^{-1}, $\beta = 100$, the size of the moving box are $d_x = d_y = d_z = 0.035$ m.

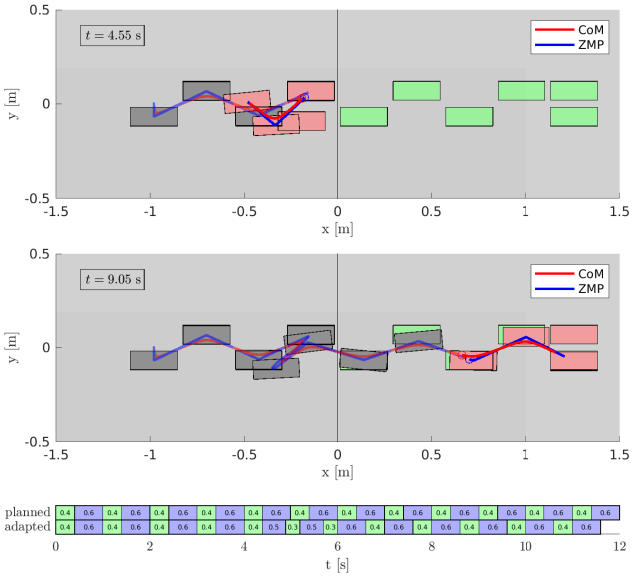

Fig. 7. V-FAPA in the *2-patches* scenario. The robot is walking in a straight line and is pushed at time 4.5 s (slightly before the first snapshot). Now the robot is allowed to adapt the footstep position to the other patch, and is able to tolerate a stronger push.

$F = 3$, $\boldsymbol{v}_1 = (0.28, 0.13)^T$ m, $\boldsymbol{v}_2 = (0.2, 0.43)^T$ m, $\boldsymbol{v}_3 = (-0.12, 0.43)^T$ m, $\boldsymbol{v}_4 = (-0.2, 0.13)^T$ m, $\Delta_z^m = -0.10$ m, $\Delta_z^M = 0.10$ m, $\Delta_\theta^m = -0.4$ rad, $\Delta_\theta^M = 0.4$ rad, $T_{\text{ds}}^{\min} = 0.3$ s, $T_{\text{ds}}^{\max} = 0.5$ s, $T_{\text{ss}}^{\min} = 0.5$ s, $T_{\text{ss}}^{\max} = 0.7$ s, $t_{\text{change}} = 0.1$ s, $M = 100$, $w_x = w_y = w_z = w_\theta = w_{\text{ds}} = w_{\text{ss}} = 1.0$ and $w_b = 0.01$. Simulation videos are available at `https://youtu.be/4_QYsZH1E7Y`.

Simulations take place in 3 different scenarios: *empty*, which is completely flat with no obstacles, and is represented using a single patch; *2-patches* is constituted by two patches at different heights (0 and 0.06 m); *stairs* has a total of 7 patches of increasing height. While walking, the robot is subject to impulsive pushes (lasting 0.01 s), transformed in equivalent acceleration imparted on the CoM.

Algorithm	Solver	Average [s]	Std dev. [s]	Max [s]
F-FAPA	`ipopt`	0.0207	0.0041	0.0467
F-FAPA	`knitro`	0.0144	0.0032	0.0329
V-FAPA	`bonmin`	0.3164	0.2075	1.2098
V-FAPA	`knitro`	0.0316	0.0393	0.3985

TABLE I

PERFORMANCE METRICS OF F-FAPA IN THE *empty* SCENARIO AND V-FAPA IN THE *2-patches* SCENARIO, USING DIFFERENT SOLVERS.

In the first simulation, the robot is walking forward in the *empty* scenario. At 4.5 s it receives a 15.6 m/s^2 push in the direction $(-2, -1, 0)$, that without FAPA would make the MPC infeasible. F-FAPA reacts by adapting footstep positions, orientations and timings concurrently, allowing the MPC to recover feasibility. Figure 5 shows nominal and adapted footsteps, trajectories and step timings.

In the second simulation (shown in Fig. 6), the scenario is *2-patches*, and the robot must climb a step. Upon receiving the push, the footsteps do not change significantly, because the F-FAPA algorithm is not allowed to move the footstep to the other patch. As a result, the tolerable push is smaller, i.e., 7.8 m/s^2.

In the third simulation (shown in Fig. 7), the scenario is still *2-patches*, but now the scheme is using V-FAPA. When the push is perceived, the first predicted footstep is moved to the lower patch, and as a result the increase of the tolerable push intensity is very significant, i.e., the same as in the *empty* scenario.

In the last simulation, the robot is moving through a more complex environment constituted by a long staircase. While climbing, the robot is subject to multiple pushes, triggering several footstep adjustments. Figure 1 shows a stroboscopic view of the motion.

To discuss the real-time applicability of the scheme, we report performance metrics in Table I. The solvers used are `ipopt` and `knitro` for F-FAPA, and `bonmin` and `knitro` for V-FAPA. `knitro` is faster overall, but `ipopt` still demonstrates good performance for F-FAPA, compatible with real-time requirements. For V-FAPA, `bonmin` is clearly too slow, while `knitro` has an average performance that is real-time on average, but some outliers violate the requirements. Since all results in this paper are simulated, real-time performance is desirable but not critical. However, it is necessary for hardware implementation, which is why we will be working to guarantee real-time performance in future works.

VII. CONCLUSIONS

We presented a module for adapting positions, orientations and timings in such a way to enhance our IS-MPC scheme, using a gait feasibility constraint. Simulated results show that the plan is adapted in a very flexible way in reaction to strong pushes. In our MATLAB prototype, the performance is fully compatible with real time in the case of F-FAPA, while not yet in the case of V-FAPA. We believe that an optimized C++ implementation will be able to meet real-time requirements. Future work will be aimed at fully accommodating these requirements, as well as including the high-level planner [15] inside the architecture so that global replanning is possile.

REFERENCES

[1] A. Herdt, N. Perrin, and P. B. Wieber, "Walking without thinking about it," in *2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010, pp. 190–195.

[2] M. R. O. A. Maximo and R. J. M. Afonso, "Mixed-integer quadratic programming for automatic walking footstep placement, duration, and rotation," *Optimal Control Applications and Methods*, vol. 41, no. 6, pp. 1928–1963, 2020.

[3] N. Bohórquez and P.-B. Wieber, "Adaptive step duration in biped walking: A robust approach to nonlinear constraints," in *17th IEEE-RAS Int. Conf. on Humanoid Robots*, 2017, pp. 724–729.

[4] S. Caron and Q.-C. Pham, "When to make a step? tackling the timing problem in multi-contact locomotion by topp-mpc," in *17th IEEE-RAS Int. Conf. on Humanoid Robots*, 2017, pp. 522–528.

[5] A. Ibanez, P. Bidaud, and V. Padois, "Emergence of humanoid walking behaviors from mixed-integer model predictive control," in *2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 4014–4021.

[6] F. M. Smaldone, N. Scianca, L. Lanari, and G. Oriolo, "Feasibility-driven step timing adaptation for robust MPC-based gait generation in humanoids," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1582–1589, 2021.

[7] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti, "Walking control based on step timing adaptation," *IEEE IEEE Trans. on Robotics*, vol. 36, no. 3, pp. 629–643, 2020.

[8] M. Naveau, M. Kudruss, O. Stasse, C. Kirches, K. Mombaur, and P. Souères, "A reactive walking pattern generator based on nonlinear model predictive control," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 10–17, 2017.

[9] N. Bohórquez and P.-B. Wieber, "Adaptive step rotation in biped walking," in *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2018, pp. 720–725.

[10] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2018.

[11] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS Int. Conf. on Humanoid Robots*, 2014, pp. 279–286.

[12] N. Scianca, D. De Simone, L. Lanari, and G. Oriolo, "MPC for humanoid gait generation: Stability and feasibility," *IEEE Trans. on Robotics*, vol. 36, no. 4, pp. 1171–1178, 2020.

[13] A. S. Habib, F. M. Smaldone, N. Scianca, L. Lanari, and G. Oriolo, "Handling non-convex constraints in mpc-based humanoid gait generation," in *2022 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2022, pp. 13 167–13 173.

[14] J. Ding, X. Xiao, and N. Tsagarakis, "Nonlinear optimization of step duration and step location," in *2019 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2019, pp. 2259–2265.

[15] M. Cipriano, P. Ferrari, N. Scianca, L. Lanari, and G. Oriolo, "Humanoid motion generation in a world of stairs," *Robotics and Autonomous Systems*, vol. 168, p. 104495, 2023.

[16] S. Caron, A. Escande, L. Lanari, and B. Mallein, "Capturability-based pattern generation for walking with variable height," *arXiv:1801.07022*, 2018.

[17] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control based on divergent component of motion," *IEEE Trans. on Robotics*, vol. 31, no. 2, pp. 355–368, 2015.

[18] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *6th IEEE-RAS Int. Conf. on Humanoid Robots*, 2006, pp. 200–207.

[19] T. Sugihara, Y. Nakamura, and H. Inoue, "Real-time humanoid motion generation through ZMP manipulation based on inverted pendulum control," in *2002 IEEE Int. Conf. on Robotics and Automation*, vol. 2, 2002, pp. 1404–1409.

[20] A. Zamparelli, N. Scianca, L. Lanari, and G. Oriolo, "Humanoid gait generation on uneven ground using intrinsically stable MPC," *IFAC-PapersOnLine*, vol. 51, pp. 393–398, 2018.

[21] R. J. Afonso, M. R. O. A. Maximo, and R. K. Galvão, "Task allocation and trajectory planning for multiple agents in the presence of obstacle and connectivity constraints with mixed-integer linear programming," *Int. Journal of Robust and Nonlinear Control*, vol. 30, no. 14, pp. 5464–5491, 2020.