Accelerating Model Predictive Control for Legged Robots through Distributed Optimization

Lorenzo Amatucci[1,2], Giulio Turrisi[1], Angelo Bratta[1], Victor Barasuol[1], Claudio Semini[1]

Abstract— **This paper presents a novel approach to enhance Model Predictive Control (MPC) for legged robots through Distributed Optimization. Our method focuses on decomposing the robot dynamics into smaller, parallelizable subsystems, and utilizing the Alternating Direction Method of Multipliers (ADMM) to ensure consensus among them. Each subsystem is managed by its own Optimal Control Problem, with ADMM facilitating consistency between their optimizations. This approach not only decreases the computational time but also allows for effective scaling with more complex robot configurations, facilitating the integration of additional subsystems such as articulated arms on a quadruped robot. We demonstrate, through numerical evaluations, the convergence of our approach on two systems with increasing complexity. In addition, we showcase that our approach converges towards the same solution when compared to a state-of-the-art centralized whole-body MPC implementation. Moreover, we quantitatively compare the computational efficiency of our method to the centralized approach, revealing up to a 75% reduction in computational time. Overall, our approach offers a promising avenue for accelerating MPC solutions for legged robots, paving the way for more effective utilization of the computational performance of modern hardware. Accompanying video at `https://youtu.be/Yar4W-Vlh2A`. The related code can be found at `https://github.com/iit-DLSLab/DWMPC`**
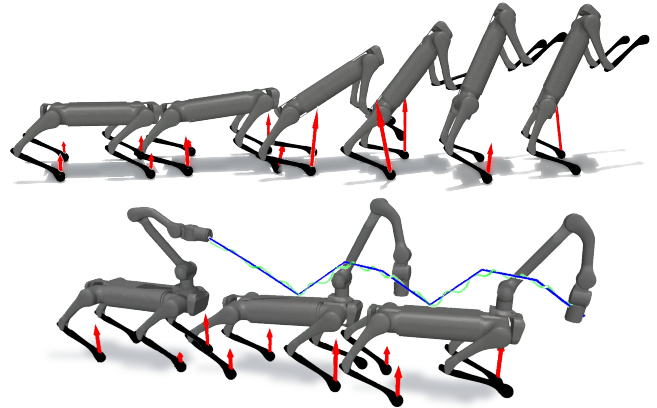
Fig. 1. Simulation snapshots of robotic systems controlled by the proposed MPC with distributed optimization, performing different agile motions. On the top, a quadruped standing up on two feet and walking forward. On the bottom, a quadruped manipulator following a triangular spiral with the manipulator end-effector, the reference is highlighted in blue while the actual trajectory is in green.

I. INTRODUCTION

Planning and control for legged systems presents significant challenges due to their inherently nonlinear dynamics. Designing trajectories for such systems is a demanding task, as these trajectories must satisfy system constraints, including the robot's kinematic limits, actuator torque limits, and the constraints imposed by the surrounding environment, such as obstacles or gaps. Model Predictive Control (MPC) [1] has emerged as a powerful technique for generating complex motions in robotic systems, particularly in the domain of legged robots [2]. MPC approaches calculate the optimal control inputs based on the modeled system, resulting in complex constrained optimization problems that need to be solved rapidly enough to achieve online re-planning of the optimal trajectory. Recent advancements in onboard hardware computational power and the development of more efficient Nonlinear Programming (NLP) solvers like [3], [4], and [5] have opened doors to solving efficiently complex Optimal Control Problem (OCP). However, the solution time is still a crucial factor to be considered when designing a control architecture for legged systems. To mitigate the computational complexity associated with the whole-body

dynamics of the robot and speed up the solving time of the OCP, the works [6], [7], and [8] opted to approximate the robot with the centroidal model [9]. This approach reduces the robot's dynamics to its center of mass (CoM) motion, neglecting the influence of individual limbs on the base. Although this approximation may seem simplistic, it considers the key dynamic effects on the system. Additionally, it facilitates the integration of feasibility constraints, such as contact friction. The centroidal model has demonstrated effectiveness in stabilizing locomotion for both quadrupeds and bipeds, although it is necessary to carefully craft reference trajectories to track agile motions.

To reduce the drawbacks of such a model, [10] and [11] developed an MPC formulation based on the Kino-Dynamic model, which takes into account joint position and velocity, neglects the effect of limb acceleration on the base dynamics while maintaining the effect of the robot joint configuration on the inertia matrix. As a result, the Kino-Dynamic model avoids the highly non-linear terms while simultaneously providing a better approximation than the centroidal model. In contrast, [12] and [13] used the full-body model, entirely exploiting the robot's capabilities. The complexity of the OCPs considered in those implementations is remarkable, given the limited time budget available for solving them in the feedback MPC scheme. Indeed, it needs to be considered that the computational complexity of the NLP implementa-

[1] Dynamic Legged Systems Laboratory, Istituto Italiano di Tecnologia (IIT), Genova, Italy. Email: `name.surname@iit.it`
[2] Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS), Università di Genova, Genova, Italy.

tion increases with the number of state variables in the OCP, typically following a cubic $O(n^3)$ law [14]. This poses a significant limitation on the number of variables that can be feasibly included in the optimization process and restricts the length of the MPC horizon. Despite the continuous growth of the number of CPU cores of each generation of modern processors, the algorithms presented earlier face limitations in terms of parallelizability. This limits the effective utilization of the computational resources and restricts how the solver's solution time scales with the number of decision variables. A different paradigm that tries to reduce the computation burden is presented in [15]. The authors worked towards a reduction of the computational effort by decoupling a quadruped robot into two bipeds. This brought a consistent reduction in the number of optimization variables that drastically decreased the computational time; nevertheless, the formulation they employed limits the approach to only the offline generation of periodic orbits, too slow to be computed online in a feedback MPC scheme. In contrast, [16] proposed a similar decoupling principle and applied it to quadratic programming-based nonlinear controllers. While the parallelizability of their approach leads to remarkable performances, particularly in terms of disturbance rejection properties inherited from the decoupling framework, their implementation requires the knowledge of the centralized system optimum. The approach is thus limited to tracking pre-computed periodic orbits. Finally, [17] developed local controllers based on the dynamic decomposition from [15], which can stabilize the centralized system across various scenarios. However, their approach is not suitable for generating highly dynamic motions due to the intrinsic limitations of instantaneous controllers. The above-mentioned methods also lack easy adaptability to more complex systems, such as a quadruped robot with one or several articulated arms on top.

Our approach divides the robot into multiple subsystems while ensuring coherence between the solutions of the decoupled systems with a consensus formulation. In particular, we designed an easily scalable method to speed up the computation of the OCP on legged systems exploiting a parallelized implementation of the Alternating Direction Method of Multipliers (ADMM). It is worth highlighting that ADMM has already been successfully used in legged robotics by [18]. Their work primarily focused on rigorously enforcing consensus between simplified and whole-body optimizations within a two-step optimization framework, which is common in locomotion controllers. In contrast, our approach focuses on decomposing the robot dynamics into reduced subsystems and using ADMM to ensure consensus among them. In other words, our approach boils down to running in parallel a separate MPC for each subsystem we have divided our robot into, using consensus ADMM to maintain consistency between the optimizations. This allows our algorithm to scale effectively with more complex robot configurations. For instance, the issue of integrating an articulated arm onto a quadruped is simplified to merely adding another subsystem in parallel to the others and adding the relative
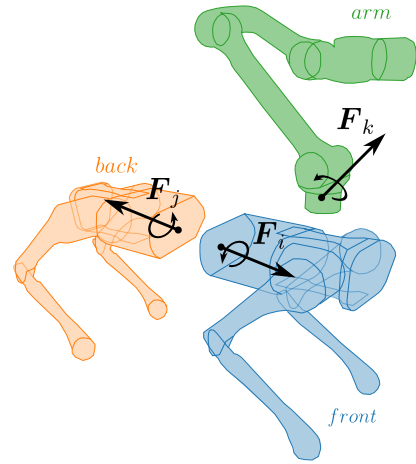

Fig. 2. Schematic of a quadruped robot with an articulated arm on top split into three separate subsystems. Each section of the robot sees the dynamic effect of the other parts through the interaction wrench $\boldsymbol{F}$.

consensus. By adopting this methodology, our method is able to solve complex whole-body motions (Fig. 1), overcoming at the same time the curse of dimensionality that is normally associated with the use of the robot's full dynamics.

A. Contributions

The main contributions of this work are:
- the development of a novel scalable framework based on ADMM that reduces the computational time of the OCP problem, e.g. two times reduction for a quadruped robot and up to four times for a quadruped plus a 6 Degrees of Freedom (DoF) arm on top, Fig. 2.
- a numerical evaluation of the scalability of the proposed approach to systems of increasing complexity, e.g. adding an arm on a quadruped robot does not affect the computational time.
- a quantitative evaluation, in simulation of the performance and convergence property of the proposed method against a state-of-the-art implementation of the same whole-body MPC problem.

B. Outline

This paper is organized as follows. Section II presents the necessary backgrounds on the ADMM algorithm and MPC. Section III describes our approach in detail. Section IV-A demonstrates the convergence property of our formulation, while Section IV-B shows the performance in terms of computational time of our distributed whole body MPC, in a simulation environment. Finally, Section V draws the final considerations and conclusions.

II. BACKGROUND

A. Alternating Direction Method of Multipliers

ADMM is an efficient approach for solving optimization problems. It has been broadly adopted in both robotics and machine learning fields [19], [20]. The ADMM algorithm solves problems in the form:

$$\min_{\boldsymbol{w},\boldsymbol{z}} \quad \boldsymbol{f}(\boldsymbol{w}) + \boldsymbol{g}(\boldsymbol{z})$$
$$\text{s.t.} \quad \boldsymbol{A}\boldsymbol{w} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{c} \tag{1}$$

Where $\boldsymbol{A}$,$\boldsymbol{B}$, and $\boldsymbol{c}$ are respectively the matrices and the vector that define the equality constraint. $\boldsymbol{w}$ and $\boldsymbol{z}$ are two separate decision variables. $\boldsymbol{f}(\boldsymbol{w})$ and $\boldsymbol{g}(\boldsymbol{z})$ are the two objectives of the cost function. The augmented Lagrangian for (1) is:

$$\mathcal{L}_\rho(\boldsymbol{w},\boldsymbol{z},\boldsymbol{y}) = \boldsymbol{f}(\boldsymbol{w}) + \boldsymbol{g}(\boldsymbol{z}) + \boldsymbol{y}^T(\boldsymbol{A}\boldsymbol{w} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}) + \frac{\rho}{2}||\boldsymbol{A}\boldsymbol{w} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c}||^2 \quad (2)$$

where $\boldsymbol{y}$ is the dual variable associated with the equality constraint and ρ is the penalty parameter related to the constraint violation. The ADMM exploits the separation of the cost terms between the $\boldsymbol{w}$ and $\boldsymbol{z}$ decision variables to split the problem and thus tackling smaller and simpler sub-problems at each iteration [19]. The solution to (1) can be found iterating till convergence the following:

$$\boldsymbol{w}^{n+1} = \min_{\boldsymbol{w}} \mathcal{L}^\rho(\boldsymbol{w}, \boldsymbol{z}^n, \boldsymbol{y}^n)$$
$$\boldsymbol{z}^{n+1} = \min_{\boldsymbol{z}} \mathcal{L}_\rho(\boldsymbol{w}^{n+1}, \boldsymbol{z}, \boldsymbol{y}^n) \quad (3)$$
$$\boldsymbol{y}^{n+1} = \boldsymbol{y}^n + \rho(\boldsymbol{A}\boldsymbol{w} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{c})$$

where the superfixed n indicates the iteration number. In the first two steps of (3), we update the two primal variables $\boldsymbol{w}$ and $\boldsymbol{z}$ sequentially, and then we calculate the dual variable; this results in a sequential implementation of the algorithm. A widely adopted approach performs the updates of primal variables, $\boldsymbol{w^{n+1}}$ and $\boldsymbol{z^{n+1}}$ in parallel. Parallelization is achieved at the expense of both the convergence rate and convergence guarantees, prioritizing on the other hand computational speed. [21] showed that including additional regularization terms in the cost partially recovers convergence rate and guarantees. In our method, we will use a special case of ADMM called *consensus* ADMM [19], in which the problem is in the form:

$$\min_{\bar{\boldsymbol{w}},\boldsymbol{w}_1,\boldsymbol{w}_2,\dots,\boldsymbol{w}_{N_{sys}}} \sum_{i=1}^{N_{sys}} \boldsymbol{f}_i(\boldsymbol{w}_i) \quad (4)$$
$$\text{s.t.} \quad \boldsymbol{w}_i = \bar{\boldsymbol{w}}_i, \quad i = 1, \dots, N_{sys}$$

where N_{sys} is the number of subsystems we split the main problem into. $\boldsymbol{w}_i$ is a local copy of a subvector of the global decision variable $\bar{\boldsymbol{w}}$. The objective in this formulation is to optimize $\bar{\boldsymbol{w}}$ through the separate optimization of the $\boldsymbol{w}_i$ variables. In this way, we transform the centralized optimization involving only $\bar{\boldsymbol{w}}$ into a network of smaller local independent problems. In our implementation, we will divide the robot model into separate independent OCP, which can be solved in parallel. The decision variables and constraints of the global problem are reduced to the ones related to the considered subsystems, making the problem much smaller. The consensus is then established between the optimizations, to avoid non-physical behaviors.

B. Model Predictive Control

In this section, we will first briefly introduce the dynamic model we are using and then we will present the OCP. The efficacy of the control input derived from an MPC strategy is directly linked with the accuracy of its predictions. Carefully modeling the dynamics of the robot is necessary to maximize its performance. Therefore, we have chosen to employ the articulated rigid body dynamics [22] with hard contacts to characterize the dynamics of a legged robot accurately. We define the state of the robot at instant k by the generalized coordinates $\boldsymbol{q}_k \in R^{n_q}$ and the generalized velocity $\boldsymbol{v}_k \in R^{n_v}$. We write the equation of motion of legged robots in discrete time following a semi-implicit Euler scheme as:

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k - \boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{S}\boldsymbol{\tau} - \boldsymbol{J}^T\boldsymbol{\lambda})dt$$
$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k \oplus \boldsymbol{v}_{k+1}dt \quad (5)$$

where $\boldsymbol{M} \in R^{n_v \times n_v}$ and $\boldsymbol{b} \in R^{n_v}$ are respectively the mass matrix and the "bias" vector that includes the Coriolis, centrifugal, and gravitational terms. The dependency of $\boldsymbol{M}$ and $\boldsymbol{b}$ from $\boldsymbol{q}_k$ and $\boldsymbol{v}_k$ has been dropped in favor of readability. The variable $\boldsymbol{\tau} \in R^{n_u}$ corresponds to the joint torque, while $\boldsymbol{S} \in R^{n_v \times n_u}$ is a selector matrix. Matrix $\boldsymbol{J} \in R^{n_v \times n_c}$ is the stack of jacobian associated with each contact and $\boldsymbol{\lambda} \in R^{n_c}$ is the stack of ground reaction forces. Considering the dynamics described before, we can write an MPC that solves the following OCP in a receding horizon fashion:

$$\min_{\boldsymbol{x},\boldsymbol{u}} \quad \boldsymbol{l}_T(\boldsymbol{x}_N) + \sum_{k=0}^{N-1} \boldsymbol{l}(\boldsymbol{x}_k, \boldsymbol{u}_k) \quad (6a)$$
$$s.t. \quad \boldsymbol{v}_{k+1} = \boldsymbol{v}_k - \boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{S}\boldsymbol{\tau} - \boldsymbol{J}^T\boldsymbol{\lambda})dt \quad (6b)$$
$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k \oplus \boldsymbol{v}_{k+1}dt \quad (6c)$$
$$\boldsymbol{J}\boldsymbol{v}_k = \boldsymbol{v}_e \quad (6d)$$
$$\boldsymbol{p}_k|_z - \boldsymbol{p_{ref}}_k|_z = 0 \quad (6e)$$
$$\boldsymbol{u}_k \in \text{U}_k \quad (6f)$$
$$\boldsymbol{x}_k \in \text{X}_k \quad (6g)$$
$$\boldsymbol{x}_0 = \hat{\boldsymbol{x}}_0 \quad (6h)$$

where $\boldsymbol{x}_k = [\boldsymbol{v}_k, \boldsymbol{q}_k]^T \in R^{n_v+n_q}$ and $\boldsymbol{u}_k = [\boldsymbol{\tau}_k, \boldsymbol{\lambda}_k]^T \in R^{n_u+n_c}$ are respectively the state and control input. $\boldsymbol{l}(\boldsymbol{x}(\cdot))$ is a quadratic cost including a tracking term and a regularization one. $\boldsymbol{J}\boldsymbol{v}_k = \boldsymbol{v}_e$ slacks the non-slipping condition for each contact to avoid numerical instability [23]. $\boldsymbol{p}_k|_z$ is the position of the foot along the z-axis and it is constrained to guarantee that the legs in stance are touching the ground. U_k is the set of feasible control input limits by the friction cone and torque limits at the joint. The set X_k represents the feasible states constrained by the joint kinematic limits. $\hat{\boldsymbol{x}}_0$ is the state initial condition.

The described nonlinear optimization problem poses significant challenges when used for controlling legged robots since, due to their complexity, they are difficult to solve at high frequencies. Even with the latest advancements highlighted in [12], the update rate of the final MPC is limited to 50Hz. Our approach aims to enhance the capabilities of modern solvers by diving the OCP into independent problems with a reduced number of decision variables that are solved in parallel. For instance, decoupling a quadruped into two parts, as illustrated in Fig. 2, results in a reduction

with respect to the centralized approach by 25% of the number of variables in each subsystem's optimization. While the number of variables of the system incorporating also the additional arm on top is reduced by 40%. More details on the computational time gains derived from this reduction are provided in Section IV-B.

III. DISTRIBUTED WHOLE BODY MPC

The main contribution of this work is to leverage the efficiency and parallelizability of the consensus ADMM to speed up the solution time of the NLP in (6). Our method provides a framework for partitioning the original OCP into smaller, independent problems while ensuring consensus among the solutions. Through this approach, we can significantly decrease the computational time while augmenting our system's complexity, such as incorporating an articulated arm onto a quadruped robot, without significantly impacting the solution time, as reported in Section IV-B. We can effectively decouple the robot into distinct subsystems, as illustrated in the example in Fig. 2. The consensus ADMM implementation is used to enforce the consistency of the dynamic between the different subsystems and prevent them from drifting apart. Without consensus, we will only have decoupled independent systems that are no longer solving together the centralized OCP in (6).

A. Decomposed dynamics

The decomposition of the robot dynamics is based on the idea of considering the separated subsystems interacting with each other through a wrench $\boldsymbol{F}$. For the ith subsystem the wrench $\boldsymbol{F}_i$ synthesizes the dynamic effect of the other j subsystems that we are splitting the robot into. The equations of motion of the ith subsystem also considering the rigid body connection constraint are written as:

$$\boldsymbol{v}_{i,k+1} = \boldsymbol{v}_{i,k} - \boldsymbol{M}_i^{-1}(\boldsymbol{b}_i - \boldsymbol{S}\boldsymbol{\tau}_i - \boldsymbol{J}_i^T\boldsymbol{\lambda}_i - \boldsymbol{J}_{F,i}^T\boldsymbol{F}_i)dt \tag{7a}$$

$$\boldsymbol{J}_{F,i}\boldsymbol{v}_{i,k} = \boldsymbol{J}_{F,j}\boldsymbol{v}_{j,k} \quad \forall j \neq i \tag{7b}$$

for $i, j \in \mathcal{X}$ where $\mathcal{X}$ is the set of subsystems we have divided our robot into. $\boldsymbol{J}_{F,i}$ is the Jacobian that maps generalized velocity of the subsystems into the velocity at the interface with the other systems. Manipulating (7) we can write $\boldsymbol{F}_i$ as:

$$\boldsymbol{F}_i dt = (\bar{\boldsymbol{J}}_F\bar{\boldsymbol{M}}^{-1}\bar{\boldsymbol{J}}_F^T)^{-1}\Delta v_{tot,k} \tag{8}$$

Here, $\bar{\boldsymbol{M}} = diag(M_0, \ldots, M_{N_{sys}})$ and $\bar{\boldsymbol{J}}_F = \begin{bmatrix} \boldsymbol{J}_{F,0} \ldots \boldsymbol{J}_{F,N_{sys}} \end{bmatrix}^T$. Δv_{tot} represent the sum of the effect of all systems and is defined as:

$$\Delta v_{tot} \equiv \sum_{*\in\mathcal{X}} \boldsymbol{J}_{F,*}\left(\boldsymbol{v}_{*,k} + \boldsymbol{M}_*^{-1}(\boldsymbol{b}_* - \boldsymbol{S}\boldsymbol{\tau}_* - \boldsymbol{J}_*^T\boldsymbol{\lambda}_*)dt\right) \tag{9}$$

It is worth noting that Δv_{tot} is obtained by applying the dynamic of (7) to each subsystem composing the considered body. The obtained coupling wrench in (8) depends on all subsystem's generalized velocity and coordinates. Given the parallelization employed in this work, it is not possible to

have the jth subsystems variables beforehand. This forced us to utilize the past information at algorithm iteration n when calculating the $n+1$ update for the ith subsystem as done in the ADMM. Such an approximation, deriving from the subsystem decoupling, does not affect the optimization performances, as empirically demonstrated in section IV-A. Indeed, such an approximation is comparable to the one already used in centralized MPC employing Newton-base NLP solvers, which linearized at each iteration based on the previous optimal solution [24]. Finally, the subsystem dynamics in (7a) can be rewritten explicating the dependency of $\boldsymbol{F}_i$ as:

$$\boldsymbol{v}_{i,k+1}^{n+1} = \boldsymbol{v}_{i,k}^{n+1} - \boldsymbol{M}_i^{-1}(\boldsymbol{b}_i - \boldsymbol{S}\boldsymbol{\tau}_i^{n+1} - \boldsymbol{J}_i^T\boldsymbol{\lambda}_i^{n+1} + \\ -\boldsymbol{J}_{F,i}^T\boldsymbol{F}(\boldsymbol{q}_i^{n+1}, \boldsymbol{v}_i^{n+1}, \boldsymbol{\lambda}_i^{n+1}, \boldsymbol{\tau}_i^{n+1}, \boldsymbol{q}_j^n, \boldsymbol{v}_j^n, \boldsymbol{\lambda}_j^n, \boldsymbol{\tau}_j^n))dt \tag{10}$$

Equation (10) now only depends on the ith system current decision variables, while the jth subsystem's influence only appears as a parameter coming from the last n^{th} iteration. The system dynamics are reduced to only the floating base plus the joints considered in the ith subsystem. We have lost the direct dependency on the other joints, but we gained a model of reduced size that can be utilized in our distributed MPC.

B. Distributed optimization

In this subsection, we will delve into the formulation of the distributed OCP. The peculiarity of such implementation lies in the fact that we are going to solve (6) defining N_{sys} optimization problems, one per each subsystem we are dividing the robot into. Such problems are smaller than the problem of (6), and can be solved in parallel, thus speeding up the computation. Each subsystem optimization problem is defined utilizing the system dynamic in (10) and the constraint (7b). Such constraint serves to ensure coherence between velocities at the interface of the subsystems, thereby preventing them from drifting apart. Since we are decoupling the subsystems to enable parallel optimization, analogously to what we did for the decomposed dynamics, (7b) cannot be directly used, but it can be formulated as a consensus problem. Defining the residual between the subsystems i and j at the time k as :

$$\boldsymbol{r}_{i,j} = \boldsymbol{J}_{F,i}\boldsymbol{v}_i - \boldsymbol{J}_{F,j}\boldsymbol{v}_j \tag{11}$$

it is possible to formulate each single subsystem optimization. We take the consensus ADMM implementation illustrated in section II-A and tailor it for the OCP in (6). For convenience, we utilized the scaled dual variable $\bar{y} = y/\rho$. In this way, the Lagrangian in (2) is reduced to its quadratic term. We defined the cost for the subsystem i as:

$$\Phi^i = \boldsymbol{l}_T(\boldsymbol{x}_i(N)) + \sum_{k=0}^{N-1}\left[\boldsymbol{l}(\boldsymbol{x}_{k,i}, \boldsymbol{u}_{k,i}) + \\ + \sum_{i,j\in\mathcal{X}} ||\boldsymbol{r}_{i,j} - \bar{\boldsymbol{y}}_{i,j}^n||_\rho^2\right] + ||\boldsymbol{x}_i - \boldsymbol{x}^n||_\sigma^2 \tag{12}$$

where the first two terms are the components of the original cost in (6) that depend on the ith subsystem decision
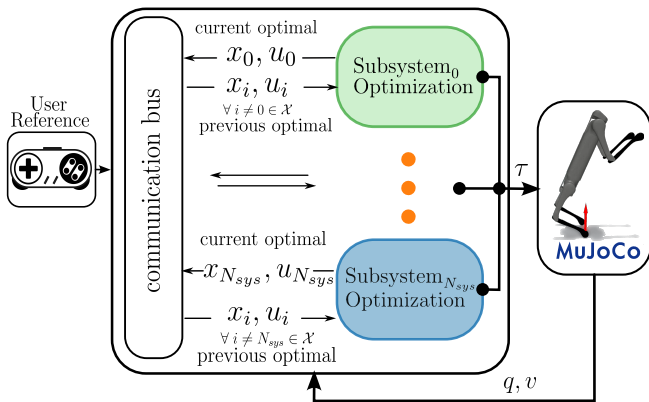
Fig. 3. Block scheme of the proposed control framework highlighting the communication bus between the decomposed Subsystem Optimizations.

Algorithm 1 Distributed Whole-body Optimization

> **Data**: x^{guess}, u^{guess}
> **repeat**
>   **for** subsystems **do** in *parallel*
> x_i^n, u_i^n ⟵ **Map**(x^{guess}, u^{guess})
> x_i^{n+1}, u_i^{n+1} ⟵ **SubSystemSolver**(x_i^n, u_i^n) (13)
>   **end for**
> $\bar{y}^{n+1} = \bar{y}^n + r^{n+1}$
> x^{guess}, u^{guess} ⟵ x^{n+1}, u^{n+1}
> **until** stopping criterion
> **return** x^{n+1}, u^{n+1}

---

variables. ρ and σ are positive weight parameters that penalize consensus violations and regularize the solution for the whole-body. The last term of (12) introduced by [21] works as a regularizer towards the optimal values found at the previous iteration and it has proven to be crucial for increasing the numerical stability of the algorithm.

Thus, the global OCP in (6) is reduced to the iterative solution, executed in parallel, of the local OCPs:

$$\min_{x^i, u^i} \Phi^i \quad \text{s.t. eq. (10) , eqs. (6c) to (6h)} \quad (13)$$

where only the states and control corresponding to the ith subsystem appear as a decision variable. The dynamics constraint in (6b) is replaced by (10), the cost function is the one shown in (12), and the constraint form (6) relative to the subsystem's variables are passed to the local OCP. During each iteration of the distributed algorithm, we perform a single full Newton step for each subsystem optimization before updating the dual variable. An outline of the implementation is reported in Algorithm (1), where the **Map**() function only extracts the subsystem variables from the global one. As for [25] we defined the stopping criteria of our algorithm based on the l_2-norm of the residual:

$$||r^n||_2 < \epsilon \quad (14)$$

where ϵ is a positive scalar threshold on the residual. Our algorithm does not come with convergence guarantees, however, similar to other ADMM-based implementations on nonconvex problems such as those in [25] and [18], the solver has empirically proven to be reliable, as shown in Section IV-A.

C. Distributed MPC

For the online execution of our framework, instead of running the described algorithm to convergence at each control loop, we implemented a receding horizon scheme. In this scheme, each local solver receives the last optimal solution from the other subsystems and the feedback from the robot. Then, we execute one Newton step for each subsystem and apply only the torque calculated at the first step of the horizon to the robot. We then shift the rest of the solution

by one time-step and send it back to the communication bus to update the dual variable and provide the current local optimal to the other optimizations for the next control loop. A schematic of the control framework used in the simulation is presented in Fig. 3.

IV. RESULT

In this section, we present the results of different analyses to demonstrate the benefits of the proposed approach. We analyze the performance of our algorithm considering two different systems: a quadruped robot and a quadruped with a manipulator on top, which we will call a *quadruped manipulator*. In the first case, we split the robot into two parts: front and back. All subsystems include a floating base and the decision variables linked with the considered joints, e.g. the front section will consider only the front legs' joint torques, positions, velocities, and the corresponding ground reaction force. This reduces the decision variables per stage from R^{61} to R^{37}, since we are discarding the 24 states and control linked with the joints included in the other half of the robot. Thanks to this reduction in the state we can solve the local OCP faster, as exemplified by the case of the quadruped manipulator, where the distributed approach can run 4 times faster than the centralized one. Indeed, as depicted in Fig. 6, for the quadruped manipulator case, the solving time is on average 10 ms for the distributed case, while for the centralized one is 40 ms.

In the case of the quadruped manipulator, we include a third subsystem in the optimization consisting of the articulated arm, as shown in Fig. 2. In our example, the articulated arm has 6 DoF and thus it adds 18 decision variables per node (joint position, velocity, and torque for each DoF) to the centralized optimization problem. In our formulation, on the other hand, the arm is treated as an independent third optimization with 31 decision variables executed in parallel to the other two. It is worth remembering that the complexity of the optimization scales with the cube of the number of stage decision variables. Indeed, a naive centralized implementation is cursed by the increased dimension of the problem, while our implementation utilizes unexploited computational power by parallelizing the additional subsystem optimization.

In the remainder of this section, we will first focus on the convergence analysis of the proposed approach, showing
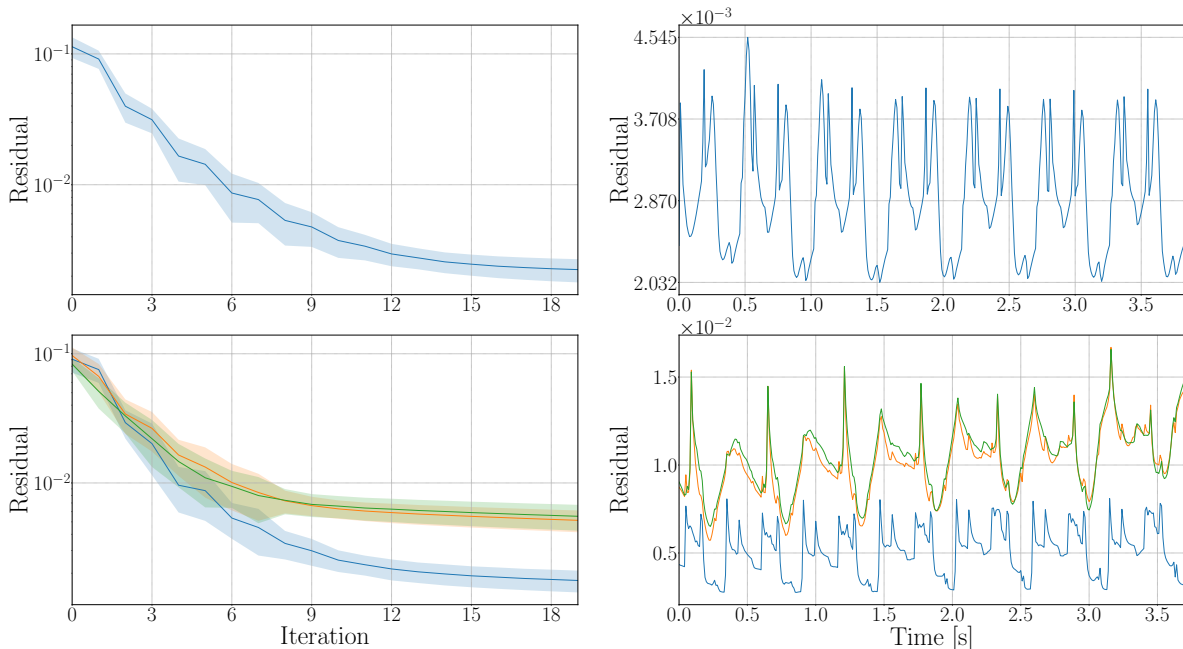
Fig. 4. The two plots on the left show the trend of the l_2 norm of the residuals along the iteration of Algorithm 1. On top is the residual for the quadruped with no arm, while on the bottom are the residuals for the quadruped manipulator. The right side plots show the time plot of the residual norm while the robot is trotting in simulation. Again, the top plot is for the robot with no arm, and the bottom one is for the quadruped manipulator.

the residual trend with both systems and comparing our distributed MPC approach with a centralized MPC implementation. Subsequently, we will demonstrate the benefits of our approach in terms of solving time and provide examples of agile maneuvers that are possible thanks to the whole-body implementation. The distributed optimization approach and the centralized one have been implemented using Adam [26], an open-source library for rigid-body dynamics and Acados [4], an open-source Sequential Quadratic Programming solver tailored for optimal control. We chose to utilize Acados for its ease of use, but our method is not tailored to any specific solver and can be reused with other NLP solvers. Both formulations utilize a prediction horizon of 50 nodes with a discretization time of 0.01 seconds, and they are both run at 50Hz for the sake of comparison. All the presented results have been obtained using an Intel Core i7 13700H laptop CPU. For the simulations, we utilized Mujoco [27] as physics engine.

A. Convergence analysis

In this subsection, we aim to empirically analyze the convergence properties and stability of our algorithm. We begin by examining the l_2 norm of the residual, as calculated in (11). This metric provides insight into the quality of the consensus achieved. Fig. 4 illustrates the trend over the iterations of Algorithm 1. As previously explained, we consider two scenarios: one with the quadruped split in half and the other with the robot plus the arm. It should be noted that in the former case, the residual is one, while in the latter case, they become 3, indicating consensus among the three subsystems. The plots depict the average over one hundred simulations, with the system initialized in different config-

urations and with randomized warm starts. The variance is represented by the colored area around the mean value. As shown, the residual converges within a few iterations to small values. Notably, the value the residual reaches corresponds to an error between the subsystems at the end of the predicted trajectories in the orders of millimeters or less. In the second column of Fig. 4, we display the trend of the residual norm of the MPC. In this case, we reported how the residual value remains in the order of 10^{-3} when the distributed approach is utilized in a receding horizon fashion. The data was recorded while the robot was trotting in simulation following a desired speed of $0.3\frac{m}{s}$.

Finally, in Fig. 5 we compare, in simulation, the centralized and the distributed approach, reporting the phase plot for each joint. The figure illustrates that the steady-state solutions of both methods converge close to the same values, depicting how our solver response is the same as one from a centralized implementation.

B. Performance evaluation

In this subsection, we show the main benefits of our implementation. As displayed in Fig. 6 we compared the solving time of the distributed and centralized implementation for the quadruped and quadruped manipulator. We showed the occurrence of the solving times recorded in a 5 second simulation reporting in the histogram the relative frequencies. In simulation, the robot is asked to follow a desired velocity of $0.3m/s$ and is perturbed repeatedly by an external force of random direction and magnitude in the range between $\pm30N$ for 0.1s each time. The graph shows that the solving times for the distributed controller are on average half of the ones of the centralized implementation for the quadruped
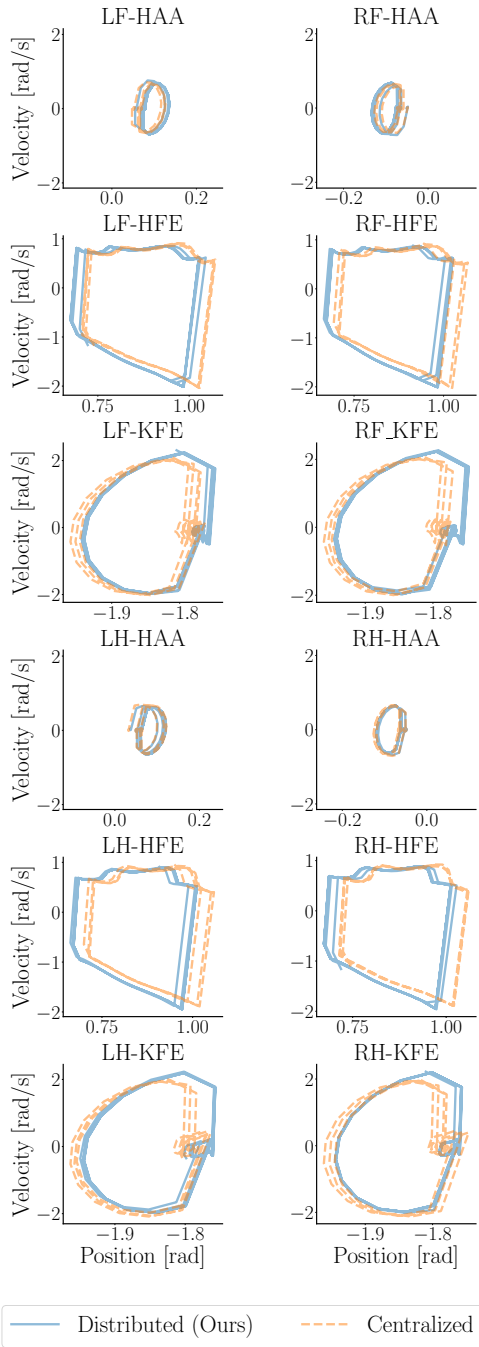
Fig. 5. Phase plots of the robot trotting with the centralized (dashed blue line) and distributed (orange line) solutions. The plot shows a recording of the robot trotting in simulation at a desired speed of 0.3 m/s. Where RF-LF-LH-RH stands for Right Front, Left Front, Left Hind, Right Hind and, HAA, HFE, KFE stand for Hip Abduction/Adduction, Hip Flexion/Extension, and Knee Flexion/Extension.

robot while for the quadruped manipulator, the solving time is reduced by a factor of four.

It is worth noting that our implementation is not affected by the increased complexity of the system. Adding the arm on the quadruped robot does not increase the computational time. It should be mentioned that the times reported for the distributed implementation are the one of the slowest subsystem optimizations. This consideration remains valid,
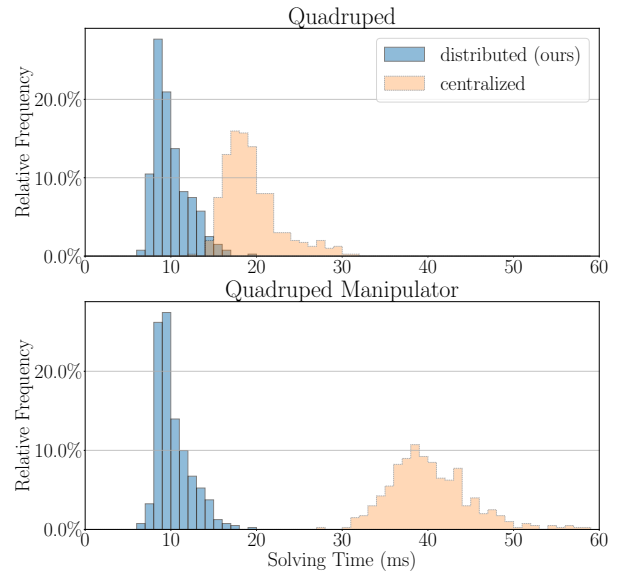

Fig. 6. Relative frequencies for the occurrence of the solution time of the receding horizon problem in (6) recorded over one hundred simulations. In blue is our distributed implementation while in orange is the centralized one. On the top, the solution refers to the quadruped robot model while on the bottom plot the quadruped plus the arm.

as long as the new system we want to include can be considered as an additional subsystem that can be processed in parallel as it has been done for the arm. As already mentioned, even though the reported times refer to our implementation using Acados, changing the solver at the core of our implementation will not affect the relative gains, but only the absolute numbers in the same way for both centralized and distributed approaches.

Finally, to demonstrate the benefit of a whole-body implementation and the robustness of our formulation, we show the robot performing complex maneuvers that need whole-body coordination. In the top part of Fig. 1 we show the quadruped stand-up and walk on two feet. No particular effort is required in designing the reference trajectories since the MPC can generate and stabilize agile maneuvers starting from simple unfeasible trajectories. Indeed, for the emergence of the biped walking behavior, it was sufficient a step function for the robot's desired base pitch equal to $\pi/2$ and a reference constant velocity to the base equal to $0.2m/s$.

The bottom part of Fig. 1 shows the quadruped manipulator trotting while its end-effector is tracking a triangular spiral defined in the world frame. In this case, the only reference we are passing to the MPC is the desired spiral trajectory for the arm's end-effector.

Both simulations can be seen in the accompanying video.

V. CONCLUSION

This paper presented a novel approach to decompose the robot dynamics into reduced and more tractable subsystems to accelerate MPC for legged robots. Utilizing ADMM we ensure consensus among the parallel subsystem's optimization demonstrating significant improvements in computational efficiency. The parallelizability of our method

facilitates the integration of additional limbs such as articulated arms without compromising the solving time, and thus making the system easily scalable. Through extensive numerical simulations, we have validated the convergence and performance of our approach across two systems of varying complexity, highlighting its effectiveness compared to state-of-the-art whole-body MPC implementations. The quantitative simulations have underscored the substantial reduction in computational time; in particular, for the quadruped model we gained a reduction of two times while for the quadruped manipulator, the achieved reduction is four times. These gains derive from the fact that we have added a new subsystem to be solved in parallel with the others so the computational time for our approach remains constant, while for a centralized whole-body optimization it drastically increases due to a higher number of decision variables per node. These gains in computational time come from our more effective use of modern hardware resources.

As future work, we plan to test our implementation on hardware and include other popular nonlinear solvers such as [3] and [5] in our framework allowing to choose the most suitable solver for a given scenario. We are also going to extend our work to other robot morphologies, e.g. humanoid robots.

REFERENCES

[1] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Pub, 2022.

[2] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, 2017, pp. 577–584.

[3] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[4] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[5] F. Farshidian *et al.*, "OCS2: An open source library for optimal control of switched systems," [Online]. Available: https://github.com/leggedrobotics/ocs2.

[6] S. Hong, J.-H. Kim, and H.-W. Park, "Real-time constrained nonlinear model predictive control on so(3) for dynamic legged locomotion," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 3982–3989.

[7] G. Romualdi, S. Dafarra, G. L'Erario, I. Sorrentino, S. Traversaro, and D. Pucci, "Online non-linear centroidal mpc for humanoid robot locomotion with step adjustment," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 412–10 419.

[8] N. Rathod, A. Bratta, M. Focchi, M. Zanon, O. Villarreal, C. Semini, and A. Bemporad, "Model predictive control with environment adaptation for legged locomotion," *IEEE Access*, vol. 9, 2021.

[9] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous robots*, vol. 35, no. 2-3, pp. 161–176, 2013.

[10] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified mpc framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.

[11] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3402–3421, 2023.

[12] C. Mastalli, S. Chhatoi, T. Corbéres, S. Tonneau, and S. Vijayakumar, "Inverse-dynamics mpc via nullspace resolution," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 3222–3241, 2023.

[13] E. Dantec, M. Naveau, P. Fernbach, N. Villa, G. Saurel, O. Stasse, M. Taix, and N. Mansard, "Whole-body model predictive control for biped locomotion on a torque-controlled humanoid robot," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 638–644.

[14] J. N. Nganga and P. M. Wensing, "Accelerating second-order differential dynamic programming for rigid-body systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7659–7666, 2021.

[15] W.-L. Ma, N. Csomay-Shanklin, and A. D. Ames, "Coupled control systems: Periodic orbit generation with application to quadrupedal locomotion," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 935–940, 2021.

[16] V. R. Kamidi, J. Kim, R. T. Fawcett, A. D. Ames, and K. Akbari Hamed, "Distributed quadratic programming-based nonlinear controllers for periodic gaits on legged robots," *IEEE Control Systems Letters*, vol. 6, pp. 2509–2514, 2022.

[17] W.-L. Ma, N. Csomay-Shanklin, S. Kolathaya, K. A. Hamed, and A. D. Ames, "Coupled control lyapunov functions for interconnected systems, with application to quadrupedal locomotion," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3761–3768, 2021.

[18] R. Budhiraja, J. Carpentier, and N. Mansard, "Dynamics consensus between centroidal and whole-body models for locomotion of legged robots," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6727–6733.

[19] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, 2011.

[20] G. Bravo-Palacios and P. M. Wensing, "Large-scale admm-based co-design of legged robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 8842–8849.

[21] W. Deng, M.-J. Lai, Z. Peng, and W. Yin, "Parallel multi-block admm with o(1/k) convergence," *Journal of Scientific Computing*, vol. 71, no. 2, pp. 712–736, 2017.

[22] R. Featherstone and K. A. Publishers, *Robot Dynamics Algorithm*. Kluwer Academic Publishers, 1987.

[23] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer Methods in Applied Mechanics and Engineering*, vol. 1, no. 1, pp. 1–16, 1972. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0045782572900187

[24] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear mpc: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[25] Z. Zhou and Y. Zhao, "Accelerated admm based trajectory optimization for legged locomotion with coupled rigid body dynamics," in *2020 American Control Conference (ACC)*, 2020, pp. 5082–5089.

[26] G. L'Erario, G. Nava, G. Romualdi, F. Bergonti, V. Razza, S. Dafarra, and D. Pucci, "Whole-body trajectory optimization for robot multi-modal locomotion," in *2022 IEEE-RAS 21st International Conference on Humanoid Robots (Humanoids)*, 2022, pp. 651–658.

[27] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 5026–5033.