Toward Faster Optimization for Legged Robots: A Distributed Approach

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

***Abstract*— This work proposes a novel approach to accelerate Model Predictive Control (MPC) for legged robots through distributed optimization. Our method decomposes the robot dynamics into smaller subsystems, utilizing the Alternating Direction Method of Multipliers to ensure consensus among them. Given this parallelization, our approach greatly decreases the computational time of the final control law compared to state-of-the-art approaches, enabling faster control loops on complex robotic systems. Through numerical evaluations, we demonstrate the convergence of our method and compare its computational efficiency against a centralized approach, showing up to a 75 % reduction in the solving time. The full paper can be found at `https://arxiv.org/abs/2403.11742`**

I. MOTIVATION

The recent advancements in computer hardware and the development of Nonlinear Programming (NLP) solvers specially tailored for optimal control like [1], [2], and [3] have opened doors to solve online complex Optimal Control Problem (OCP). Works like [4] and [5] have achieved online re-planning while using a whole-body model to fully exploit the robot's capabilities. The high complexity reached by the OCPs in those implementations is remarkable given the limited time budget available for solving them in a receiving horizon fashion. Different paradigms try to reduce the computation burden by decomposing the system dynamics, lowering in this way the problem dimension. [6] decoupled a quadruped robot into two bipeds; nevertheless, their approach is not suitable for generating highly dynamic motions due to the intrinsic limitations of their instantaneous controller. [7] proposed a similar decoupling principle but, while the performance achieved in disturbance rejection is remarkable, their implementation requires the knowledge of the centralized optimum limiting the controller to the tracking of a pre-computed periodic orbit. Our approach, on the other hand, partitions the robot into multiple independent subsystems to speed up the computation and ensures coherence among their solutions through a consensus formulation that leverages a parallelized implementation of the Alternating Direction Method of Multipliers (ADMM). In other words, our approach boils down to running in parallel a separate Model Predictive Control (MPC) for each subsystem, with ADMM ensuring consistency between their optimizations. This enables our algorithm to be virtually independent of the complexity of the systems. For example, integrating

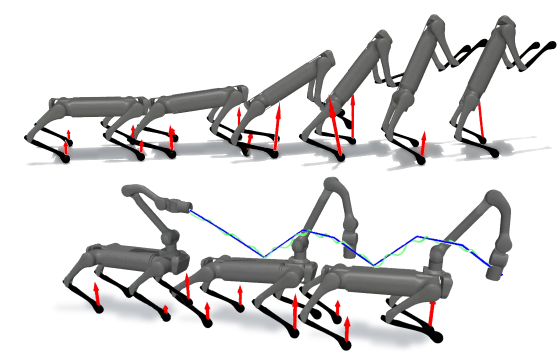

Fig. 1. Simulation snapshots of robotic systems controlled by the proposed MPC with distributed optimization, performing different agile motions. On the top, a quadruped standing up on two feet and walking forward. On the bottom, a quadruped manipulator follows a triangular spiral with the manipulator end-effector, the reference is highlighted in blue while the actual trajectory is in green. The simulation can be seen in the accompanying video*.

an articulated arm onto a quadruped only requires adding another subsystem in parallel and incorporating the relative consensus in the formulation. By adopting this methodology, our method can address complex whole-body motions (see Fig. 1), while overcoming the curse of dimensionality associated with using the robot's full dynamics.

II. DISTIBUTED WHOLE-BODY MPC

Our method runs separated optimizations in parallel, one for each subsystem we divide the robot into, followed by a unique consensus scheme. The OCP related to the i^{th} subsystem can be written as

$$\min_{x_i, u_i} \; \Phi_i \tag{1a}$$

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

$$\boldsymbol{q}_{i,k+1} = \boldsymbol{q}_{i,k} \oplus \boldsymbol{v}_{i,k+1} \tag{1c}$$

$$\boldsymbol{u}_{i,k} \in \mathrm{U}_k \text{ control constraints} \tag{1d}$$

$$\boldsymbol{x}_{i,k} \in \mathrm{X}_k \text{ state constraints} \tag{1e}$$

$$\boldsymbol{x}_{i,0} = \hat{\boldsymbol{x}}_{i,0} \text{ initial condition} \tag{1f}$$

where with $\boldsymbol{q}_{i,k} \in R^{n_q}$ we represent the generalized coordinates and with $\boldsymbol{v}_{i,k} \in R^{n_v}$ the generalized velocity. We can then define the state vector as $\boldsymbol{x}_{i,k} = \left[\boldsymbol{v}_{i,k}, \boldsymbol{q}_{i,k}\right]^T \in R^{n_v+n_q}$, and the control vector as $\boldsymbol{u}_{i,k} = \left[\boldsymbol{\tau}_{i,k}, \boldsymbol{\lambda}_{i,k}\right]^T \in R^{n_u+n_c}$, with $\boldsymbol{\tau} \in R^{n_u}$ the joint torques and $\boldsymbol{\lambda} \in R^{n_c}$ the ground reaction forces. Eq. (1b) defines the i^{th} part of the decomposed robot dynamics, 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. $\boldsymbol{S} \in R^{n_v \times n_u}$ is a selector matrix and

[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.

* Accompanying video `https://www.youtube.com/watch?v=0KcTnGYjJPw`

$\boldsymbol{J} \in R^{n_v \times n_c}$ is the stack of jacobian associated with each contact.

Our decomposition is based on the idea of considering the separated subsystems interacting with each other through a wrench $\boldsymbol{F}$. For the i^{th} subsystem, the wrench $\boldsymbol{F}_i$ synthesizes the dynamic effect of the other j subsystems and is defined as

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

Here, $\bar{\boldsymbol{M}} = diag(M_0, \ \ldots \ , M_{N_{sys}})$ and $\bar{\boldsymbol{J}}_F = \begin{bmatrix} \boldsymbol{J}_{F,0} & \ldots & \boldsymbol{J}_{F,N_{sys}} \end{bmatrix}^T$ with $\boldsymbol{J}_{F,i}$ as the Jacobian that maps the generalized velocity of the subsystems into the velocity at the conjunction point with the other systems. Δv_{tot} represents the sum of the effect of all systems and is defined as

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

Given the parallelization employed in this work, predetermining the variables of the j^{th} subsystem beforehand is not feasible. Consequently, we relied on past information at algorithm iteration n when computing the $n+1$ update for the i^{th} subsystem, analogous to the methodology employed in the ADMM. Indeed, such approximation is comparable to the one already used in centralized MPC, where Newton-based NLP solvers linearize the problem at each iteration based on the previous optimal solution [8].

Finally, the cost Φ_i is defined as:

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

in which $\boldsymbol{l}_T(\boldsymbol{x_i}(N))$ and $\boldsymbol{l}(\boldsymbol{x}_{k,i}, \boldsymbol{u}_{k.i})$ are the final and running cost that comprehend a tracking and a regularizing term. The sum $||\boldsymbol{r}_{i,j} - \bar{\boldsymbol{y}}_{i,j}^n||_\rho^2$ is the consensus term that ensures the coherence of the speeds at the interface between the subsystem, with $\boldsymbol{r}_{i,j} = \boldsymbol{J}_{F,i}\boldsymbol{v}_i - \boldsymbol{J}_{F,j}\boldsymbol{v}_j$ as the residual and $\bar{\boldsymbol{y}}_{i,j}^n$ as the scaled dual variable. Finally, $||\boldsymbol{x}_i - \boldsymbol{x}^n||_\sigma^2$ is a regularization between iterations. In each iteration of our algorithm, we perform in parallel only one full Newton step of the single subsystem optimization and then we update the dual variable. We keep iterating till a convergence criterion on the l_2-norm of the residual is not satisfied. This formulation does not come with convergence guarantees, however, the solver has empirically proven to be reliable, as shown in Fig. 3. The reader may refer to the full paper [9] for a more complete description of the overall method.

III. Result

The performance of our algorithm is analyzed using two different systems: a quadruped robot and a quadruped manipulator, where the latter includes a manipulator mounted on top of the robot's trunk. For the first system, we divide it into two parts: front and back. Each subsystem includes a floating base and only the decision variables related to their specific joints, we can then solve two problems with 37 decision variables in parallel instead of one with 61.

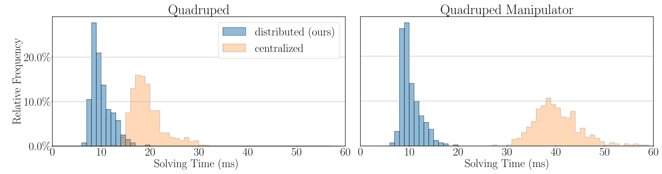

Fig. 2. Relative frequencies for the occurrence of the solution time of the receding horizon problem recorded over one hundred simulations. In blue is our distributed implementation while in orange is the centralized one. On the left, the solution refers to the quadruped robot model while on the right plot the quadruped plus the arm.

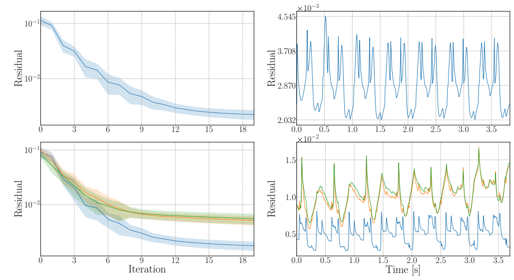

Fig. 3. The two plots on the left show the trend of the l_2 norm of the residuals along the iteration of our algorithm. On top is the residual for the quadruped with no arm, while on the bottom are the residuals for the quadruped manipulator. The right side plots show the time plot of the residual norm 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.

For the quadruped manipulator, we add a third subsystem of only 31 decision variables per stage in parallel that considers only the arm and its floating base instead of the full system with 79. This formulation accelerates the local OCP solution, as demonstrated in Fig. 2. Our distributed approach runs two times faster than the centralized one for the quadruped and four times faster for the quadruped manipulator. We empirically analyze the convergence properties and stability of our algorithm by examining the l_2 norm of the residual, since this metric provides insight into the quality of the consensus achieved. Fig. 3 shows that the residual converges in a few iterations of our algorithm to values that correspond to a position error at the end of the prediction in the orders of millimeters. Fig. 3 also displays the trend of the residual when the optimization is used in a receding horizon fashion to control the robot. Further results are presented in the full paper [9].

IV. Conclusion

In this work, we proposed a novel approach to accelerate MPC for legged robots by dividing the locomotion optimization problem into smaller, parallelizable subsystems. Utilizing a consensus ADMM implementation, we significantly improved the computational efficiency while ensuring coherence among parallel optimizations. Our method enables seamless integration of additional limbs, such as articulated arms, without compromising solving time, making the optimization easily scalable. We compared to a centralized whole-body MPC implementations, showcasing reduced computational time. Future work includes hardware validation, integration of other nonlinear solvers, and extension to other robot morphologies, e.g. humanoid robots.

REFERENCES

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] L. Amatucci, G. Turrisi, A. Bratta, V. Barasuol, and C. Semini, "Accelerating model predictive control for legged robots through distributed optimization," https://arxiv.org/abs/2403.11742, 2024.