Stagewise Implementations of Sequential Quadratic Programming for Model-Predictive Control

Armand Jordana, Sébastien Kleff, Avadesh Meduri, Justin Carpentier, Nicolas Mansard, Ludovic Righetti

Stagewise Implementations of Sequential Quadratic Programming for Model-Predictive Control

Armand Jordana*,1, Sébastien Kleff*,1, Avadesh Meduri*,1,
Justin Carpentier[2], Nicolas Mansard[3] and Ludovic Righetti[1]

Abstract— **The promise of model-predictive control in robotics has led to extensive development of efficient numerical optimal control solvers in line with differential dynamic programming because it exploits the sparsity induced by time. In this work, we argue that this effervescence has hidden the fact that sparsity can be equally exploited by standard nonlinear optimization. In particular, we show how a tailored implementation of sequential quadratic programming achieves state-of-the-art model-predictive control. Then, we clarify the connections between popular algorithms from the robotics community and well-established optimization techniques. Further, the sequential quadratic program formulation naturally encompasses the constrained case, a notoriously difficult problem in the robotics community. Specifically, we show that it only requires a sparsity-exploiting implementation of a state-of-the-art quadratic programming solver. We illustrate the validity of this approach in a comparative study and experiments on a torque-controlled manipulator. To the best of our knowledge, this is the first demonstration of nonlinear model-predictive control with arbitrary constraints on real hardware.**

I. INTRODUCTION

A. Motivation

Model Predictive Control (MPC) has become popular for online robot decision-making. It has shown compelling results with all kinds of robots ranging from industrial manipulators [1], quadrupeds [2]–[4] to humanoids [5], [6]. The general idea of MPC is to formulate the robot motion generation problem as a numerical optimization problem, i.e., a finite horizon Optimal Control Problem (OCP), and solve it online at every control cycle using the current state measurement as the initial state. This receding horizon strategy allows us to adapt the robot behavior as the state of the system and environment change.

In robotics, Differential Dynamic Programming (DDP) [7] is a popular choice to solve OCPs because it exploits the problem's structure well. This advantage has led to a bustling algorithmic development over the past two decades [8]–[20]. In light of the increasing number of variations of DDP, one might naively ask: why not use well-established optimization algorithms [21]? Is there anything special in MPC that cannot be tackled by, for example, an efficient implementation of Sequential Quadratic Programming (SQP) [22]? In this work, we show that special implementations of numerical methods developed by the optimization-based control community [23]–[26] are, in fact, sufficient to achieve state-of-the-art MPC on real robots.

B. Related work

Mayne first introduced DDP [7] as an efficient algorithm to solve nonlinear OCPs by iteratively applying a backward pass over the time horizon and a nonlinear forward rollout of the dynamics. This algorithm notably exhibits linear complexity in the time horizon and local quadratic convergence [27]. More recently, Todorov revived the interest in DDP by proposing the iterative Linear Quadratic Regulator (iLQR) [8], a variant discarding the second-order terms of the dynamics. It has since gained a lot of traction within the robotics community [3]–[5], [14], [18], and its similarity to Gauss-Newton optimization has been established [9], [28]. However, this approach faces two main limitations: 1) as a single shooting method, it requires a dynamically feasible initial guess, which makes the algorithm difficult to warm-start, an essential requirement to reduce computation times [12] and 2) enforcing equality and inequality constraints is not straightforward. The common practice is to enforce constraints softly using penalty terms in the cost function. But this approach is heuristic (i.e., it requires cost weight tuning) and tends to cause numerical issues [29].

Multiple shooting for optimal control, introduced in [30], addresses the first limitation: it accepts an infeasible initial guess. Several multiple shooting variants of DDP/iLQR were proposed in [12], [14] with significantly improved convergence abilities, which have enabled nonlinear MPC at high frequency on real robots [1], [3], [6], [14].

The second issue of enforcing constraints inside a DDP-like algorithm has been addressed in several works. [10] uses a DDP-based projected Newton method to bound control inputs. This approach has further been improved and deployed on a real quadruped robot in [17]. More recently, augmented Lagrangian methods have been used to enforce constraints in iLQR/DDP algorithms [11], [13], [16], [19]. However, their convergence behavior is less understood than DDP, whose seminal paper [7] was followed by sophisticated proofs [27]. To the best of our knowledge, it has not yet been shown that those recent DDP-based algorithms exhibit global convergence (i.e., convergence from any initial point to a stationary point) and quadratic local convergence.

This work was in part supported by the National Science Foundation grants 1932187, 2026479, 2222815 and 2315396.

* Equal contribution - first authors listed in alphabetical order.

[1] Machines in Motion Laboratory, New York University, USA `aj2988@nyu.edu`, `sk8001@nyu.edu`, `am9789@nyu.edu`, `lr114@nyu.edu`

[2] Inria - Département d'Informatique de l'École normale supérieure, PSL Research University. `justin.carpentier@inria.fr`

[3] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse. `nmansard@laas.fr`

Besides, an open topic of debate is whether to use a nonlinear or a linear rollout to enforce dynamics constraints in the forward pass. Previous work on single shooting methods has argued that a nonlinear rollout can reduce the number of iterations [31], while [28], [32] showed mixed results both experimentally and theoretically. Recently, a nonlinear rollout was implemented by a popular multiple-shooting variant of iLQR [14]. However, we are unaware of any comparisons between linear and nonlinear rollouts in the context of multiple shooting methods.

In the face of these challenges, we propose to take a fresh look at the earlier literature. Indeed, Dunn et al. showed that Newton's method could also be implemented in a DDP-like fashion and equally benefit from linear complexity in the time horizon and quadratic convergence [33]. This finding indicates that optimal control does not fundamentally require new nonlinear optimization tools but only tailored implementations that exploit the time-induced sparsity structure. This naturally led to numerous extensions to the constrained case. For example, Wright proposed to use an SQP formulation with an active set to address arbitrary nonlinear constraints [22]. However, because of the limitations of active set methods, Wright focused on the single shooting case resolution. Then, Pantoja et al. proposed an efficient implementation of SQP for control inequality constraints [34]. Di Pillo et al. proposed a tailored implementation of a Quasi-Newton method with an augmented Lagrangian-based approach [35]. Dohrmann et al. studied equality-SQP for optimal control [36]. Additionally, [37]–[41] studied the tailored implementation of Interior Point Methods for optimal control.

In the late 2000's, with the promise of high-frequency linear MPC, a large body of work studied how to tailor Quadratic Programming (QP) for the Constrained Linear Quadratic Regulator (CLQR) problem. This was done with active set methods [42], [43], ADMM-based solvers [23] and interior point method [25], [44]. We refer the reader to [24] for an extensive survey.

To solve the nonlinear case, [26] recently proposed efficient software with an SQP implementation for OCP. [45] studied how to exploit the sparsity induced by time in IPOPT [46]. Unfortunately, this line of work has not benefited from as much experimental study as DDP-like algorithms. Recently, [47] showed impressive experimental results on quadrupeds using a tailored SQP implementation based on HPIPM [25], which lies in the continuity of previous works using [25], [26] on real hardware [48]–[50]. However, to the best of our knowledge, we are not aware of closed-loop constrained nonlinear MPC on torque-controlled robots.

Recently, [51] reviewed the models and optimization algorithms used in robotics in the context of MPC. As emphasized in this survey, we argue that there is a gap between the optimization-based control and the robotics community. On the one hand, an important part of the robotics community followed the successes of [52] and continued to propose DDP-like algorithms. On the other hand, the optimization-based control community followed the work of [22], [34] and proposed efficient implementations of established optimization algorithms [24], [26], [45]. In this work, we aim to bridge this gap.

C. Contributions

In this paper, we follow the line of thought of the optimization-based control community in order to push the limits of closed-loop nonlinear MPC in robotics. First, we shed light on the direct connection between modern multiple-shooting DDP-like algorithms and textbook SQP algorithms. Second, we show through an experimental study that a standard stagewise SQP formulation is, in fact, superior to the state-of-the-art FDDP [14]. Third, we re-implement a QP solver tailored for optimal control by leveraging Riccati recursions in order to maintain linear complexity in the time horizon while also enforcing constraints [23], [53]. Using this custom QP implementation inside the SQP formulation, we can solve arbitrary nonlinear constrained OCPs efficiently while inheriting the well-known convergence properties of standard SQPs. Lastly, we demonstrate the ability of this SQP formulation to enforce arbitrary nonlinear constraints in MPC experiments on a torque-controlled manipulator. To the best of our knowledge, this is the first demonstration of closed-loop nonlinear MPC with hard constraints on real hardware. The optimization software has been open-sourced and integrated with Crocoddyl [14] to ensure reproducible experiments and enable easy use by the research community.

II. Sequential Quadratic Programming for Optimal Control Problems

Notations. The derivative of a function f with respect to a vector v is denoted by $\nabla_v f$, similarly for second order derivatives with respect to vectors u, v is denoted as $\nabla^2_{uv} f$. Bold characters are used to denote a sequence of vectors indexed over a time horizon, e.g., $\mathbf{v} = \{v_1, \cdots, v_k, \cdots\}$.

In this work, we study constrained optimal control problems of the form:

$$\min_{\boldsymbol{x},\boldsymbol{u}} \sum_{k=0}^{T-1} \ell_k(x_k, u_k) + \ell_T(x_T) \tag{1a}$$

$$\text{subject to} \quad x_{k+1} = f_k(x_k, u_k), \tag{1b}$$

$$c_k(x_k, u_k) \geq 0, \qquad 0 \leq k < T, \tag{1c}$$

$$c_T(x_T) \geq 0, \tag{1d}$$

where $\boldsymbol{x} = (x_1, ..., x_T)$ is the state sequence, $\boldsymbol{u} = (u_0, u_1, ..., u_{T-1})$ the control inputs and x_0 is the initial state provided by the user (e.g., estimated state). The transition functions, f_k, the cost functions, ℓ_k and the constraint functions, c_k, are supposed to be $\mathcal{C}^2$. To keep notations simple, we formulated the OCP with inequality, which also encompasses equality constraints. Yet, equality constraints can be treated separately from general inequalities in practice.

Considering the state sequence as an optimization variable allows the optimization procedure to iterate through infeasible trajectories, yielding a **multiple-shooting** approach [30].

Remark 1. *One may choose to optimize only on the sequence of control inputs and define the dynamics constraints implicitly, yielding a* ***single shooting*** *approach. In that case, if no inequality constraints are considered, the problem is unconstrained and can be solved via an efficient implementation of Newton's method [33] or with a modified Newton's method with a nonlinear rollout, namely DDP [7].*

In this work, we focus on multiple-shooting approaches. The associated Lagrangian is:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \ell_T(x_T) - \mu_T^T c_T(x_T) + \sum_{k=0}^{T-1} \ell_k(x_k, u_k) - \lambda_{k+1}^T(x_{k+1} - f_k(x_k, u_k)) - \mu_k^T c_k(x_k, u_k), \tag{2}$$

where $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_T)^T$ and $\boldsymbol{\mu} = (\mu_0, ..., \mu_T)^T$ are Lagrange multipliers.

Intuitively, SQPs iteratively solve QPs to find a tuple $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ that satisfies the KKT conditions [54]. Note that we consider the constraint $x_0 = \bar{x}_0$ to be implicit. Therefore, no Lagrange multiplier is associated with that constraint. For simplicity, we slightly abuse the notation by referring to $l_0(x_0, u_0)$ instead of $l_0(u_0)$ even if x_0 is fixed.

Remark 2. *Note that one might also want to consider a generic constraint on the initial condition $c(x_0)$ (e.g., in the context of robust control [55]). To do so, the initial condition must be considered an optimization variable. Although we do not consider this setting to clarify derivations, our work naturally extends to this case.*

More precisely, at the n^{th} iteration, given a guess on the tuple $(\boldsymbol{x}^{[n]}, \boldsymbol{u}^{[n]}, \boldsymbol{\lambda}^{[n]}, \boldsymbol{\mu}^{[n]})$, we aim to find a correction on the guess by solving the following QP problem (Algorithm 18.1, [21]):

$$\min_{\Delta \boldsymbol{x}, \Delta \boldsymbol{u}} \sum_{k=0}^{T-1} \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix}^T \begin{bmatrix} Q_k & S_k \\ S_k^T & R_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^T \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \frac{1}{2} \Delta x_T^T Q_T \Delta x_T + \Delta x_T^T q_T \tag{3a}$$

$$\text{s.t.} \quad \Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k + \gamma_{k+1}, \tag{3b}$$

$$D_k \Delta x_k + E_k \Delta u_k + c_k \geq 0, \qquad 0 \leq k < T. \tag{3c}$$

$$D_T \Delta x_T + c_T \geq 0. \tag{3d}$$

where $q_T = \nabla_x \ell_T(x_T^{[n]})$ and

$$\begin{aligned} Q_T &= \left(\nabla_{xx}^2 \ell_T - \mu_T^T \nabla_{xx}^2 c_T\right)(x_T^{[n]}) \\ Q_k &= \left(\nabla_{xx}^2 \ell_k + \lambda_{k+1}^T \nabla_{xx}^2 f_k - \mu_k^T \nabla_{xx}^2 c_k\right)(x_k^{[n]}, u_k^{[n]}) \\ S_k &= \left(\nabla_{xu}^2 \ell_k + \lambda_{k+1}^T \nabla_{xu}^2 f_k - \mu_k^T \nabla_{xu}^2 c_k\right)(x_k^{[n]}, u_k^{[n]}) \\ R_k &= \left(\nabla_{uu}^2 \ell_k + \lambda_{k+1}^T \nabla_{uu}^2 f_k - \mu_k^T \nabla_{uu}^2 c_k\right)(x_k^{[n]}, u_k^{[n]}) \end{aligned} \tag{4}$$

$$\begin{aligned} A_k &= \nabla_x f_k(x_k^{[n]}, u_k^{[n]}), \quad B_k = \nabla_u f_k(x_k^{[n]}, u_k^{[n]}) \\ q_k &= \nabla_x \ell_k(x_k^{[n]}, u_k^{[n]}), \quad r_k = \nabla_u \ell_k(x_k^{[n]}, u_k^{[n]}) \quad 0 \leq k < T. \end{aligned} \tag{5}$$

Q_k, S_k, R_k are the Hessian of the Lagrangian with respect to the state and control inputs, D_k, E_k, D_T are the constraint Jacobian and $\gamma_{k+1} = f_k(x_k^{[n]}, u_k^{[n]}) - x_{k+1}^{[n]}$ is the constraint violation which is often referred to as dynamic gaps [14] or defects [12]. Note that this formulation of SQP and its associated QP is precisely the same as the one proposed in the seminal multiple-shooting article [30].

Remark 3. *In his seminal work introducing multiple shooting [30], Bock exploits the sparsity of the quadratic problem of Eq.* (3) *induced by the multiple shooting structure yielding to a cubic complexity in the time horizon, T. Instead, this work exploits the sparsity induced by time in order to achieve a linear complexity.*

The solution of the QP $(\Delta \boldsymbol{x}, \Delta \boldsymbol{u})$ is then used to update the guess:

$$\boldsymbol{x}^{[n+1]} = \boldsymbol{x}^{[n]} + \alpha \Delta \boldsymbol{x} \tag{6a}$$

$$\boldsymbol{u}^{[n+1]} = \boldsymbol{u}^{[n]} + \alpha \Delta \boldsymbol{u} \tag{6b}$$

Here, α is the step size and is chosen using a line search method. $\boldsymbol{\lambda}^{[n+1]}$ and $\boldsymbol{\mu}^{[n+1]}$ are then replaced by the associated Lagrange multiplier of the QP (3) [54]. Provided assumptions on the Problem (1), SQPs can guarantee local quadratic convergence or super-linear convergence in the case of quasi-Newton approximation [54].

III. Solving the Unconstrained QP

In this section, we review the case when the OCP has no constraints to recall the equivalence between the backward Riccati recursions used in DDP and a special type of Gaussian elimination for tridiagonal matrices, specialized to the sparsity pattern of the KKT system arising in OCPs. This result, insufficiently known in the robotics community, will then be exploited to propose a novel extension to the constrained case.

Proposition 1. *Without inequality constraints, the KKT conditions of Problem* (3) *can be written as a block tri-diagonal symmetric matrix equation.*

$$\begin{bmatrix} \Gamma_1 & M_1^T & 0 & 0 & \cdots & 0 \\ M_1 & \Gamma_2 & M_2^T & 0 & \cdots & 0 \\ 0 & M_2 & \Gamma_3 & M_3^T & \cdots & 0 \\ 0 & 0 & M_3 & \Gamma_4 & \ddots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \ddots & \Gamma_T \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ \vdots \\ s_T \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \\ \vdots \\ g_T \end{bmatrix} \tag{7}$$

where:

$$\Gamma_k = \begin{bmatrix} R_{k-1} & 0 & -B_{k-1}^T \\ 0 & Q_k & I \\ -B_{k-1} & I & 0 \end{bmatrix}, \quad M_k = \begin{bmatrix} 0 & S_k^T & 0 \\ 0 & 0 & 0 \\ 0 & -A_k & 0 \end{bmatrix}$$

$$\text{and} \quad s_k = \begin{bmatrix} \Delta u_{k-1} \\ \Delta x_k \\ -\lambda_k \end{bmatrix}, \quad g_k = \begin{bmatrix} -r_{k-1} \\ -q_k \\ \gamma_k \end{bmatrix}. \tag{8}$$

The proof is only computational and is detailed in the supplementary material[1]. Note that we slightly abuse the notation by using the λ as the Lagrange multiplier for both the QP and the nonlinear problem. The most straightforward way to solve such a system is to apply Gaussian elimination by using the well-known Thomas algorithm [56] (Algorithm 1) to achieve a complexity in $O(T)$.

Algorithm 1: Thomas algorithm

$\bar{\Gamma}_T \leftarrow \Gamma_T$

$\bar{g}_T \leftarrow \Gamma_T^{-1} g_T$

/* backward pass */

for $k \leftarrow 1$ *to* $T-1$ **do**

  $\bar{\Gamma}_k \leftarrow \Gamma_k - M_k^T \bar{\Gamma}_{k+1}^{-1} M_k$

  $\bar{g}_k \leftarrow \bar{\Gamma}_k^{-1}(g_k - M_k^T \bar{g}_{k+1})$

/* forward pass */

$s_1 \leftarrow \bar{g}_1$

for $k \leftarrow 1$ *to* $T-1$ **do**

  $s_{k+1} \leftarrow \bar{g}_{k+1} - \bar{\Gamma}_{k+1}^{-1} M_k s_k$

In particular, we use the knowledge of the structure (sparsity pattern) in both Γ_k and M_k to recover simpler recursions. This leads to another way of obtaining the backward Riccati equations of LQR, which is accepted knowledge in the optimization-based control community [22], [24], [33], [57] but is largely ignored in robotics.

Proposition 2. *By applying the Thomas algorithm, we recover the well-known Riccati recursions. Specifically, the* ***backward pass*** *can be done by initializing $V_T = Q_T$ and $v_T = q_T$, and then by applying the following equations:*

$$\begin{aligned} h_k &= r_k + B_k^T(v_{k+1} + V_{k+1}\gamma_{k+1}) \\ G_k &= S_k^T + B_n^T V_{k+1} A_k \qquad K_k = -H_k^{-1} G_k \\ H_k &= R_k + B_k^T V_{k+1} B_k \qquad k_k = -H_k^{-1} h_k \\ V_k &= Q_k + A_k^T V_{k+1} A_k - K_k^T H_k K_k \\ v_k &= q_k + K_k^T r_k + (A_k + K_k B_k)^T (v_{k+1} + V_{k+1}\gamma_{k+1}) \end{aligned} \tag{9}$$

Then, the ***forward pass*** *initializes $\Delta x_0 = 0$ and unrolls the linearized dynamics:*

$$\Delta x_{k+1} = (A_k + B_k K_k)\Delta x_k + B_k k_k + \gamma_{k+1} \tag{10a}$$

$$\Delta u_k = K_k \Delta x_k + k_k \tag{10b}$$

$$\lambda_k = V_k \Delta x_k + v_k \tag{10c}$$

Proof. The proof is mainly computational and relies on the analytical inversion of $\bar{\Gamma}_k$ in order to prove by recursion that:

$$\bar{\Gamma}_k = \begin{bmatrix} R_{k-1} & 0 & -B_{k-1}^T \\ 0 & V_k & I \\ -B_{k-1} & I & 0 \end{bmatrix} \tag{11a}$$

$$\bar{g}_k = \bar{\Gamma}_k^{-1} \begin{bmatrix} -r_{k-1} \\ -v_k \\ \gamma_k \end{bmatrix} \tag{11b}$$

In other words, the recursion on $\bar{\Gamma}_k$ and $\bar{g}_k$ can be substituted by the recursion on the value function hessian and gradient. The forward recursion can then be recovered using $s_{k+1} = \bar{g}_{k+1} - \bar{\Gamma}_{k+1}^{-1} M_k s_k$ and the analytical inversion of $\bar{\Gamma}_k$. Detailed derivations are provided in the supplementary material. □

This result shows that the backward Riccati equations used for Linear Quadratic Regulators (LQR) are an efficient technique for solving Quadratic Programs with a tri-diagonal symmetric KKT matrix. Consequently, these equations are ideal for solving OCPs with linear dynamics and quadratic costs.

Remark 4. *A vast amount of literature exists on tri-diagonal matrices. Understanding the structure of the KKT matrices inherent to OCP enables the use of any of these techniques directly. There are methods to factor those matrices or to solve them more efficiently, such tri-diagonal system. For instance, parallel cyclic reduction can be used in order to achieve a $O(\log(T))$ complexity, which might be interesting for problems with long time horizons [58]–[61].*

IV. Solving the Constrained QP

We are now in the position to extend the discussion of the previous section to the case with inequality constraints. When inequalities are added, the underlying KKT system associated with the QP defined in (3) will have the same tri-diagonal structure. Therefore, similar recursions can be used to solve it efficiently. For instance, HPIPM [25] derives efficiently an interior point method. Although we could have used this method, we chose OSQP [53] as it outperforms HPIPM on the set of benchmarks developed by [62]. Furthermore, ADMM-based methods are known to find a reasonably good solution in a few iterations [63], [64], an appealing feature for MPC applications. Finally, as originally discussed by [23], ADMM-based methods can easily be implemented sequentially. We illustrate this by deriving a tailored implementation of the modern ADMM-based solver OSQP [53] for OCP. However, we would like to highlight that the reader could choose any other desired sparsity exploiting QP solver depending on their preference.

A. Background on ADMM

Here, we follow [53], [63] to briefly summarize ADMM. Using generic notations, Problem (3) aims to solve a problem of the form:

$$\min_{v \in \mathrm{Dom}(g)} g(v) \ \text{ s.t. } \ Cv \in \mathcal{C} \tag{12}$$

Note that we use generic notations of the QP literature in this section. Hence, v denotes the optimization variable. Furthermore, C is a matrix, g is a convex function, and $\mathcal{C}$ is

[1] https://github.com/machines-in-motion/StagewiseSQP

a convex set. The ADMM updates are then:

$$\tilde{v}^{j+1} = \underset{v \in \mathrm{Dom}(g)}{\mathrm{argmin}}\; g(v) + \frac{\rho}{2}\|Cv - z^j + \rho^{-1}y^j\|_2^2 + \frac{\sigma}{2}\|v - v^j\|_2^2 \tag{13a}$$

$$\tilde{z}^{j+1} = \alpha C\tilde{v}^{j+1} + (1-\alpha)z^j \tag{13b}$$

$$v^{j+1} = \alpha\tilde{v}^{j+1} + (1-\alpha)v^j \tag{13c}$$

$$z^{j+1} = \Pi_{\mathcal{C}}\left(\tilde{z}^{j+1} + \rho^{-1}y^j\right) \tag{13d}$$

$$y^{j+1} = y^j + \rho\left(\tilde{z}^{j+1} - z^{j+1}\right) \tag{13e}$$

where $\Pi_{\mathcal{C}}$ is the projection operator on the convex set $\mathcal{C}$, ρ is the penalty parameter that encourages consensus between v and z, and $\alpha \in (0,2)$ is an over-relaxation parameter that improves convergence speed (e.g., when set between 1.5 and 1.8 [63]).

B. Tailored ADMM for optimal control

Here, we present the ADMM-based QP solver [63], which is tailored for optimal control problems. The presented algorithm is mainly inspired by [23]. We further introduce more recent techniques developed in ADMM [63] to the previous work [23] to improve solver performance. Finally, we provide an efficient implementation of the proposed solver, enabling reliable real-robot deployment.

In order to exploit the time-induced sparsity and leverage the Riccati recursions specific to the OCP formulation, we design a QP solver that always maintains the feasibility of the dynamics. More precisely, we consider the optimization variable to be

$$v \triangleq (\Delta\boldsymbol{x}, \Delta\boldsymbol{u})^T \tag{14}$$

and g to be the cumulative cost function defined in Eq. (3a), We propose to define $\mathrm{Dom}(g)$ to encode the set of feasible dynamics (3b) while $Cv \in \mathcal{C}$ encodes the path and terminal constraint (3c) (3d). Eq. (13a) can therefore be written as:

$$\begin{aligned}\min_{\Delta\boldsymbol{x},\Delta\boldsymbol{u}} &\sum_{k=0}^{T-1}\begin{bmatrix}\Delta x_k\\ \Delta u_k\end{bmatrix}^T\begin{bmatrix}Q_k & S_k\\ S_k^T & R_k\end{bmatrix}\begin{bmatrix}\Delta x_k\\ \Delta u_k\end{bmatrix} + \begin{bmatrix}q_k\\ r_k\end{bmatrix}^T\begin{bmatrix}\Delta x_k\\ \Delta u_k\end{bmatrix}\\ &+ \Delta x_T^T Q_T \Delta x_T + \Delta x_T^T q_T + \frac{\rho}{2}\left\|D_T\Delta x_T - z_T^j + \rho^{-1}y_T^j\right\|_2^2\\ &+ \sum_{k=0}^{T-1}\frac{\rho}{2}\left\|D_k\Delta x_k + E_k\Delta u_k - z_k^j + \rho^{-1}y_k^j\right\|_2^2\\ &+ \sum_{k=0}^{T}\frac{\sigma}{2}\left\|\Delta x_k - \Delta x_k^j\right\|_2^2 + \sum_{k=0}^{T-1}\frac{\sigma}{2}\left\|\Delta u_k - \Delta u_k^j\right\|_2^2 \end{aligned}\tag{15a}$$

$$\text{s.t.}\quad \Delta x_{k+1} = A_k\Delta x_k + B_k\Delta u_k + \gamma_{k+1}. \tag{15b}$$

We denote $\boldsymbol{y}^j = (y_0^j, y_1^j, ..., y_T^j)^T$ and $\boldsymbol{z}^j = (z_0^j, z_1^j, ..., z_T^j)^T$. Therefore, Eq. (13a) has the exact same structure as the unconstrained case mentioned in Section III. Note that the stagewise nature of the inequality constraints allows us to write the ρ dependent regularization terms in a block-sparse way as well. More precisely, as each inequality only depends on one time-step, the ρ dependent regularization term can be written as a stagewise sum of quadratic cost. Consequently, Eq. (13a) can be solved with a backward and forward pass similar to those of the unconstrained case, ensuring a linear complexity in the time horizon. Furthermore, due to the stagewise nature of the inequality constraint, Eq. (13b) - (13e) can be implemented recursively or in parallel. Algorithm 2 summarizes the procedure.

Algorithm 2: ADMM tailored for OCP

Input: $\Delta\boldsymbol{x}^j, \Delta\boldsymbol{u}^j, \boldsymbol{z}^j, \boldsymbol{y}^j$

```
/* Minimization problem */
```

1. Solve (15) using LQR to get $\widetilde{\Delta\boldsymbol{x}}^{j+1}, \widetilde{\Delta\boldsymbol{u}}^{j+1}$

```
/* Lagrange multiplier update */
```

2. **for** $k \leftarrow 1$ *to* $T-1$ **do**
3. $\quad \tilde{z}_k^j = \alpha(D_k\Delta x_k^{j+1} + E_k\Delta u_k^{j+1}) + (1-\alpha)z_k^j$
4. $\quad \Delta x_k^{j+1} = \alpha\widetilde{\Delta x}_k^{j+1} + (1-\alpha)\Delta x_k^j$
5. $\quad \Delta u_k^{j+1} = \alpha\widetilde{\Delta u}_k^{j+1} + (1-\alpha)\Delta u_k^j$
6. $\quad z_k^j = \max(c_k, \tilde{z}_k^j + \rho^{-1}y_k^j)$
7. $\quad y_k^{j+1} = y_k^j + \rho(\tilde{z}_k^j - z_k^{j+1})$
8. $\tilde{z}_T^j = \alpha D_T\Delta x_T^{j+1} + (1-\alpha)z_T^j$
9. $\Delta x_T^{j+1} = \alpha\widetilde{\Delta x}_T^{j+1} + (1-\alpha)\Delta x_T^j$
10. $z_T^j = \max(c_T, \tilde{z}_T^j + \rho^{-1}y_T^j)$
11. $y_T^{j+1} = y_T^j + \rho(\tilde{z}_T^j - z_T^{j+1})$

Output: $\Delta\boldsymbol{x}^{j+1}, \Delta\boldsymbol{u}^{j+1}, \boldsymbol{z}^{j+1}, \boldsymbol{y}^{j+1}$

C. Details on the QP

In our implementation, we take $\sigma = 10^{-6}$, $\alpha = 1.6$ and consider the same schedule as OSQP for the ρ-update [53]. Subsequently, the new backward recursion is only needed when ρ is updated (once in 25 ADMM iterations). This makes the algorithm highly efficient when compared to a standard QP solver and the solver proposed in [23] because the forward passes are very cheap (only matrix-vector multiplications), and the inversion in the backward pass happens very few times. Furthermore, the QP is warm-started by the solution of the problem with only equality constraint (dynamic-feasibility). This comes at the cost of Riccati recursions but dramatically reduces the total number of ADMM iterations required. Lastly, we compute a relative primal and dual tolerance after each ADMM iteration as discussed in [63]. We use these quantities as termination criteria.

V. Practical Implementation of the SQP

A. Gauss-Newton approximation

A common practice is to ignore the second-order term of the constraints as it is expensive to compute [8], [12], [14]. Subsequently, the second-order cost terms become:

$$\begin{aligned} Q_T &= \nabla_{xx}^2\ell_T(x_T^{[n]}), & Q_k &= \nabla_{xx}^2\ell_k(x_k^{[n]}, u_k^{[n]})\\ S_k &= \nabla_{xu}^2\ell_k(x_k^{[n]}, u_k^{[n]}), & R_k &= \nabla_{uu}^2\ell_k(x_k^{[n]}, u_k^{[n]}), \end{aligned}\tag{16}$$

for $0 \le k < T$. A direct consequence is that (3) is now independent of the multipliers. In this unconstrained case, the resulting SQP can be efficiently solved by sequentially constructing a QP at each iterate and solving it with LQR. This method was initially proposed as the Gauss-Newton Multiple Shooting (GNMS) method in [12]. However, the connection to SQPs was not discussed in the paper. Thus, GNMS is an efficient SQP algorithm to solve equality-constrained nonlinear optimization problems with block separable constraints and costs when the second-order terms are ignored.

Remark 5. *Note that more sophisticated schemes estimating the exact Hessians iteratively could be used as done in [30]. Furthermore, in the unconstrained case, the recent work of [20] suggests that the convergence benefits might compensate for the computational requirement of the second-order computations.*

B. Linear rollout

By nature of the SQP algorithm, the update step (6) is equivalent to making a linear rollout of the dynamics (cf. (10a)). However, a nonlinear rollout is often favored in DDP-like algorithms, e.g., in the popular Feasible-DDP algorithm [14] or in ALTRO [13]. While [7], [27] studied well the local and global convergence property of DDP, to the best of our knowledge there is no guarantee that these nonlinear rollout formulations maintain global convergence in the multiple shooting context. Hence, a theoretical analysis of algorithms such as Feasible-DDP remains to be done. In contrast, SQPs benefit from a very exhaustive theoretical analysis, convergence guarantees, and experimental validation in a variety of fields [21]. Furthermore, allowing the gaps to close in one step, as in [14], seems to go against the spirit of the original formulation of multiple shooting with SQP [30].

C. Line search

Now that we essentially use SQPs with a QP that exploits the block sparsity, we can leverage the vast literature of line search algorithms to select the right step length (Eq. (6)). We compared both a filter line-search [65] and a merit function-based line-search [21]. Let's define the total cost of a trajectory, the total gap violation, and the total constraint violation as:

$$\boldsymbol{\ell}(\boldsymbol{x}^{[j]}, \boldsymbol{u}^{[j]}) = \sum_{k=0}^{T-1} \ell_k(x_k^{[j]}, u_k^{[j]}) + \ell_T(x_T^{[j]}), \tag{17}$$

$$\boldsymbol{\gamma}(\boldsymbol{x}^{[j]}, \boldsymbol{u}^{[j]}) = \sum_{k=0}^{T-1} \left\| x_{k+1}^{[j]} - f_k(x_k^{[j]}, u_k^{[j]}) \right\|_\infty,$$

$$\boldsymbol{c}(\boldsymbol{x}^{[j]}, \boldsymbol{u}^{[j]}) = \sum_{k=0}^{T-1} \left\| \left[c_k(x_k^{[j]}, u_k^{[j]}) \right]_- \right\|_\infty + \left\| \left[c_T(x_T^{[j]}) \right]_- \right\|_\infty.$$

With the filter line search, a candidate $\boldsymbol{x}^{[n+1]}, \boldsymbol{u}^{[n+1]}$ is accepted if:

$$\begin{aligned} &\forall j \leq n, \quad \boldsymbol{\ell}(\boldsymbol{x}^{[n+1]}, \boldsymbol{u}^{[n+1]}) < \boldsymbol{\ell}(\boldsymbol{x}^{[j]}, \boldsymbol{u}^{[j]}) && \text{(18a)} \\ \text{or } &\forall j \leq n, \quad \boldsymbol{\gamma}(\boldsymbol{x}^{[n+1]}, \boldsymbol{u}^{[n+1]}) < \boldsymbol{\gamma}(\boldsymbol{x}^{[j]}, \boldsymbol{u}^{[j]}) && \text{(18b)} \\ \text{or } &\forall j \leq n, \quad \boldsymbol{c}(\boldsymbol{x}^{[n+1]}, \boldsymbol{u}^{[n+1]}) < \boldsymbol{c}(\boldsymbol{x}^{[j]}, \boldsymbol{u}^{[j]}) && \text{(18c)} \end{aligned}$$

In contrast, the merit function is of the form:

$$\boldsymbol{\ell}(\boldsymbol{x}^{[j]}, \boldsymbol{u}^{[j]}) + \mu_\gamma \boldsymbol{\gamma}(\boldsymbol{x}^{[j]}, \boldsymbol{u}^{[j]}) + \mu_c \boldsymbol{c}(\boldsymbol{x}^{[j]}, \boldsymbol{u}^{[j]}), \tag{19}$$

where μ_γ and μ_c are weights defined by the user. A step is accepted if the candidate allows a decrease in the merit function. The downside of the merit function is that the weight between each term has to be tuned, which can be cumbersome. On the other hand, we found that the filter line search yielded good performances and was more practical as no parameter tuning was required. In this filter line search, the cost values, gap norm, and constraint violation of all the previous iterates are stored and browsed. In our implementation, we provide the possibility of keeping only a restricted number of iterates in memory and refer to this parameter as the filter size. Although a filter keeping all the past estimates can help with problems that require many iterations (e.g., to meet a strict tolerance), it can reasonably be shortened or even discarded in practice for MPC applications. In our experience, one can select a filter size of 1 without significant loss in convergence.

D. Termination criteria

The solver is terminated once the infinity norm of the KKT condition is below a certain threshold. More precisely, the following termination criterion is used:

$$\begin{aligned} \left\| \nabla_x \mathcal{L}(\boldsymbol{x}^{[n]}, \boldsymbol{u}^{[n]}, \boldsymbol{\lambda}^{[n]}, \boldsymbol{\mu}^{[n]}) \right\|_\infty &\leq \epsilon_{SQP} \\ \left\| \nabla_u \mathcal{L}(\boldsymbol{x}^{[n]}, \boldsymbol{u}^{[n]}, \boldsymbol{\lambda}^{[n]}, \boldsymbol{\mu}^{[n]}) \right\|_\infty &\leq \epsilon_{SQP} \\ \left\| x_{k+1}^{[n]} - f_k(x_k^{[n]}, u_k^{[n]}) \right\|_\infty &\leq \epsilon_{SQP}, \quad 0 \leq k < T, \\ \left\| \left[c_k(x_k^{[n]}, u_k^{[n]}) \right]_- \right\|_\infty &\leq \epsilon_{SQP}, \quad 0 \leq k < T. \\ \left\| \left[c_T(x_T^{[n]}) \right]_- \right\|_\infty &\leq \epsilon_{SQP}. \end{aligned} \tag{20}$$

We use here the convergence criteria widely used in the optimization community [21]. Note that we do not employ a real-time iteration scheme [66] as we found in practice that letting the solver converge led to better experimental results than re-planning faster with a single SQP iteration.

As mentioned in [21], $\boldsymbol{\lambda}^{[n+1]}, \boldsymbol{\mu}^{[n+1]}$ are given by the Lagrange multiplier associated with (3). Note that we can compute them very efficiently due to the recursions. In the unconstrained case, we can use (10c). In the constrained case, we have,

$$\lambda_k = V_k \widetilde{\Delta x_k} + v_k \tag{21}$$

$$\mu_k = y_k \tag{22}$$

where V_k, v_k are derived from the Riccati recursions defined by Problem (15) and where $\widetilde{\Delta x_k}$, y_k are the primal and dual solutions of the QP (at time k).

VI. Experiments

In this section, we demonstrate the practical benefits of using a sparsity-exploiting SQP implementation for nonlinear MPC with and without inequality constraints. The solvers used in the benchmarks and experiments are open-sourced in the `mim_solvers` library[2], which uses `Crocoddyl` [14] as a base software. Rigid-body dynamics computations are done using the `Pinocchio` library [67] and its analytical derivatives [68]. All the benchmarks and figures presented in this paper can be reproduced using the dedicated repository `StagewiseSQP`[3].

Firstly, we benchmark the performance of our tailored SQP in the absence of constraints (Section III) against other popular methods that use non-linear rollouts. We show that our tailored SQP implementation performs as well and sometimes better than other methods to solve challenging OCPs. We further reinforce this claim by deploying the tailored SQP in high-frequency nonlinear MPC experiments on a torque-controlled manipulator.

Secondly, we demonstrate the validity of the SQP approach with our tailored ADMM implementation (Section IV) in the presence of nonlinear inequality constraints by deploying it on real hardware in MPC. Our experiments show that the solver can satisfy many nonlinear equality and inequality constraints while maintaining real-time solve rates.

A. *Unconstrained case: linear vs nonlinear rollout*

As emphasized in Section V-A, when only the dynamics constraint is present, the efficient SQP formulation boils down to the GNMS algorithm described in [12]. We propose for the first time to benchmark this algorithm against state-of-the-art optimal control solvers, namely DDP [7] and FDDP [14], on a set of difficult unconstrained OCPs, using a standard line-search procedure and termination criteria drawn from the SQP literature.

1) Benchmark Setup: We evaluate and compare the convergence of the following 4 solvers, namely DDP, FDDP using the default line-search from [14], FDDP using the proposed filter line-search of Section V-C, and our SQP, on a set of increasingly difficult randomized problems. We summarized the characteristics of these solvers (multiple or single-shooting, type of rollout, and line-search) in Table I. The classical single-shooting DDP and its multiple-shooting

	Multiple shooting	Rollout	Line-search
DDP	No	Nonlinear	Goldstein
FDDP (default LS)	Yes	Nonlinear	Heuristic [14]
FDDP (filter LS)	Yes	Nonlinear	Filter [65]
SQP	Yes	Linear	Filter [65]

TABLE I: Solver characteristics.

variant FDDP (default LS) are the ones implemented in the Crocoddyl library [14]. The FDDP (filter LS) was modified to incorporate the same filter line-search as our tailored SQP implementation. We use the largest possible filter size for both solvers, i.e., the maximum number of iterations.

To evaluate the performance of these solvers, the following OCPs are used:

- Kuka $(n_x = 14,\ n_u = 7)$: the task is to minimize the Cartesian distance to an end-effector goal ($3D$ reaching task in Cartesian space) under state (joint positions and velocities) and control (joint torques) regularization.
- Quadrotor $(n_x = 13,\ n_u = 4)$: the task is to minimize the distance to a desired pose (in $\mathbb{SE}(3)$) under state and control regularization.
- Double Pendulum $(n_x = 4,\ n_u = 1)$: the task is to minimize the distance to the upward equilibrium position under state and control regularization.
- Humanoid Taichi $(n_x = 77,\ n_u = 32)$: the task is to achieve a desired left foot pose (in $\mathbb{SE}(3)$) while maintaining balance on the right stance foot, starting from a double foot support configuration, under state and control regularization, and log-barrier state limits.

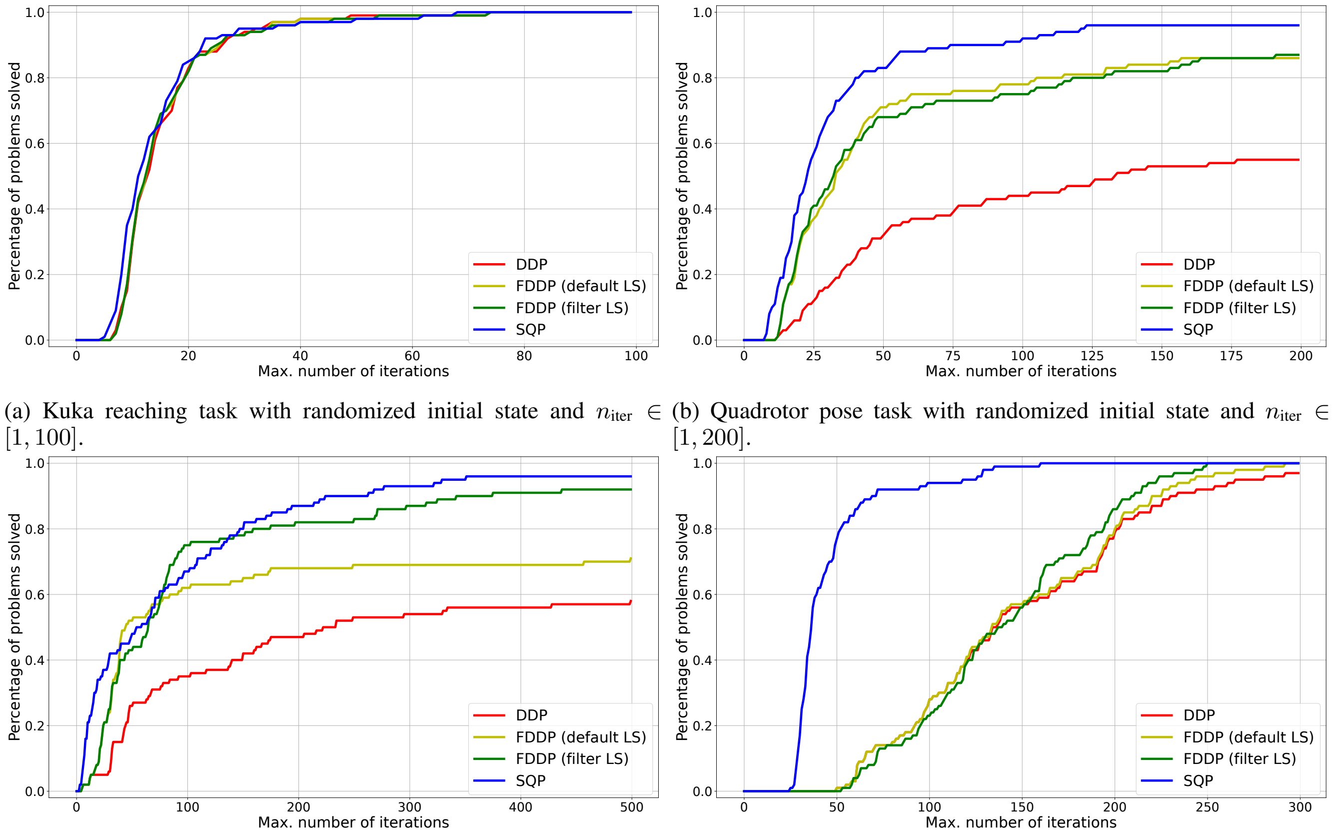
The Quadrotor, Double Pendulum, and Humanoid Taichi examples were copied from the `Crocoddyl` library. Each problem is solved for 100 randomized initial configurations or end-effector goals. The maximum number of iterations allowed n_{iter} depends on the problem. For each problem, the solvers are warm-started with the same quasi-static solution. We use the same termination criteria, the KKT residual norm (20) with tolerance set to $\epsilon_{SQP} = 10^{-4}$ on all problems. In order to reflect cases where the maximum number of iterations is hit without reaching the desired tolerance, we use as a metric the number of solved problems for a given maximum number of iterations.

2) Benchmark results: The results are shown in Figure 1. On the Kuka example, all solvers exhibit a similar behavior. The advantage of multiple-shooting over single-shooting becomes clear in the Quadrotor (Figure 1b) and Double Pendulum (Figure 1c) examples. These two examples, along with the Humanoid taichi (Figure 1d), also highlight clearly the benefit of using a filter line-search in FDDP. Most importantly, the tailored SQP solves more problems in less iterations than all the other solvers. In particular, it is clear from these benchmarks that the linear rollout has an advantage over the nonlinear one - we recall that FDDP (filter LS) (green curves) and SQP (blue curves) only differ by their rollouts (nonlinear and linear respectively).

For MPC applications, the solver's ability to converge to a desired tolerance within a small number of iterations is critical because real-time constraints impose a limited computation budget. In that respect, all experiments show that SQP is able to solve more problems in fewer iterations than all other solvers. For instance, in difficult problems like the humanoid taichi example, nearly 80% of the problems are solved by the SQP within 50 iterations, while the FDDP (filter LS) requires almost 200 iterations to solve the same amount of problems.

3) MPC experiments: We implemented FDDP (filter LS) and SQP in MPC on the KUKA LBR14 iiwa to execute the task of tracking a circle with the end-effector. The cost function includes state and torque regularization, and an end-effector position tracking term. The robotic setup follows our previous work [1], where more details are available. We used an MPC frequency of $500\,\text{Hz}$, with an OCP discretization of

[2] `https://github.com/machines-in-motion/mim_solvers`
[3] `https://github.com/machines-in-motion/StagewiseSQP`



(a) Kuka reaching task with randomized initial state and $n_{\text{iter}} \in [1, 100]$.

(b) Quadrotor pose task with randomized initial state and $n_{\text{iter}} \in [1, 200]$.

(c) Double pendulum swing-up task with randomized initial state and $n_{\text{iter}} \in [1, 500]$.

(d) Humanoid taichi task with randomized end-effector goal and $n_{\text{iter}} \in [1, 300]$.

Fig. 1: Percentage of problem solved as a function of the maximum number of iterations allowed n_{iter} on 4 randomized unconstrained OCPs for the 4 solvers: DDP, FDDP with default line-search, FDDP with filter line-search and our SQP. Our SQP exhibits a faster and more robust convergence on difficult problems, such as the humanoid taichi task.

$50\,\text{ms}$, 10 nodes in the horizon, and a maximum number of SQP iterations $n_{\text{iter}} = 5$. At each MPC cycle, the solvers are warm-started with the previous trajectory. Although both solvers exhibited equal tracking performance, they differed in their convergence speeds which corroborates our benchmarks. Indeed the cumulative costs achieved are similar but the SQP converges faster to its optimal solution as shown in Figure 2.

B. Constrained case

We now discuss the performance of the SQP with the proposed tailored ADMM implementation to handle constraints. First, we demonstrate our ability to solve constrained multi-contact OCPs. Then, we show that the proposed method can be used in MPC to satisfy arbitrary constraints on real hardware.

1) Quadruped standing task with friction cones: The Solo [69] quadruped is tasked with tracking a desired CoM position while maintaining its contact forces within the friction cone, i.e.,

$$\|F_T\|_2 \leq \mu F_N \tag{23}$$

where F_T, F_N represent the tangential and normal forces, respectively. We use 250 nodes in the OCP, with a discretization of $20\,\text{ms}$. The CoM must track a circular trajectory of $13\,\text{cm}$ diameter and $0.2\,\text{rad}\,\text{s}^{-1}$ velocity. The QP absolute and relative tolerances are set to 10^{-6}, and the termination tolerance to 10^{-4}. We use the filter line-search with the maximum size of the KKT residual termination criteria as previously described. The same OCP was solved with and without constraint (23). The solver converged in 34 iterations in the unconstrained case and in 31 iterations in the constrained case. Note that the merit function with default parameter $\mu_\gamma = \mu_c = 1$ was used in this illustrative example, which seems to benefit the solver's convergence in the constrained case over the unconstrained case. The accompanying video shows the corresponding motion, and snapshots are provided in Figure 4. Figure 5 shows the ratio of tangential over normal forces of the unconstrained and constrained solutions. One can see that the task cannot be achieved without slipping when no constraint is enforced. Hence our tailored SQP implementation can enforce nonlinear inequality constraints (Lorentz cones) while also keeping the number of iterations low.

2) MPC experiments setup: We deployed our tailored SQP implementation in MPC on the KUKA robot to achieve various constrained tasks. In all the experiments, the MPC

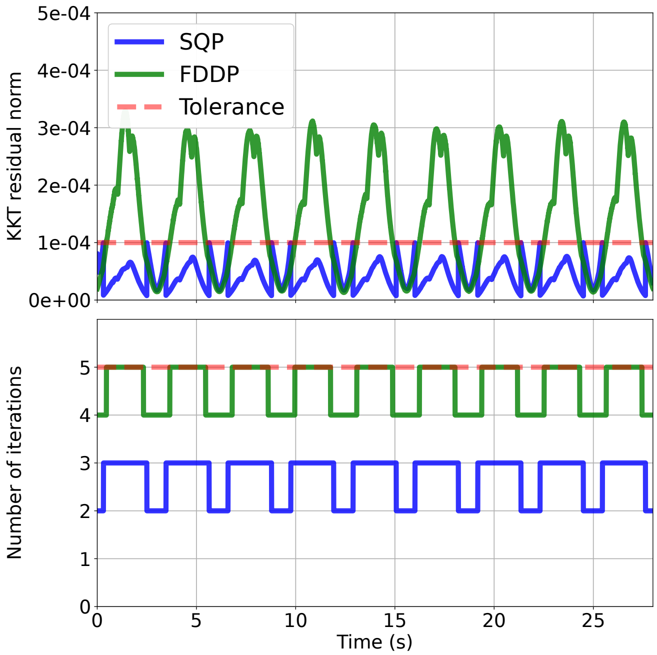


Fig. 2: KKT residual norm and number of iterations for the circle tracking task. Our SQP solver converges within 3 iterations while FDDP hits the maximum number of iterations ($n_{\text{iter}} = 5$) without reaching the desired tolerance.

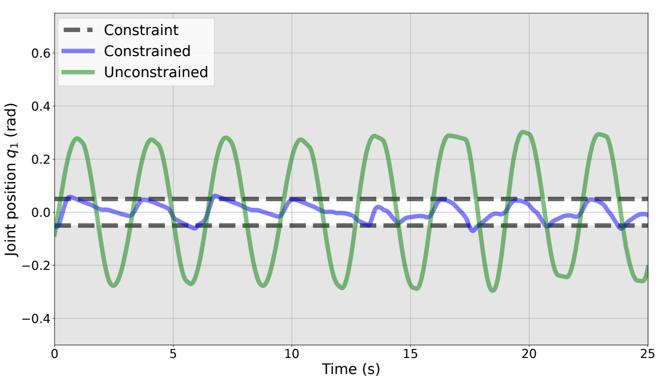


Fig. 3: Joint position q_1 for the circle task in the constrained (blue) and unconstrained case (green). The gray-shaded area represents the infeasible region.

runs at $100\,\text{Hz}$, the OCP discretization is $50\,\text{ms}$ and the horizon has 10 nodes. We allow a maximum of 4 SQP iterations and 50 QP iterations. Relative and absolute tolerances for the QP are 10^{-5} and 10^{-4} for the SQP. The size of the filter for the line-search is set to 1. The ρ penalty parameter is not reset throughout the iterations, and the primal solution is warm-started with the previous trajectory (no warm-start on the dual variables y and z which are reset to 0).

Our experiments focus on constrained circle tracking tasks: the robot must track a circle with its end-effector while satisfying various constraints in the joint space or in the end-effector space. The objective includes state and control regularization terms and a term to follow a circle with the end-effector (3D task). We observed that reducing the MPC frequency to $100\,\text{Hz}$ (vs $500\,\text{Hz}$ in the unconstrained experiments) was beneficial since it allowed to increase the maximum number of QP and SQP iterations and thereby a better convergence.

3) MPC experiments results: In the first experiment, the robot must track the circle while keeping the position of the first joint within $[-0.05\,\text{rad}, +0.05\,\text{rad}]$. The position of the constrained joint is shown in Figure 3. Without the constraint, the circle tracking average absolute error is lower ($3.3\,\text{cm}$) than the constrained case ($8.5\,\text{cm}$) but the constraint is largely violated. Increasing the end-effector tracking cost weight in the constrained case leads to an improved tracking performance but a more aggressive behavior on the robot due to an increased constraint saturation.

In the second experiment, an end-effector constraint was imposed during the circle task. Figures 6a, 6b show the Cartesian space trajectories for circle tasks in which the end-effector is constrained to lie within specific half-spaces. We observed that increasing the velocity of the reference circle had the effect of smoothing out the edges of the square. This is behavior can be explained by the prediction horizon which enables the controller to anticipate constraints: remaining some distance away from the constraint boundary is the optimal way to minimize the objective while preventing constraints violation in the future. This confirms the intuition that the horizon is crucial in constrained dynamic tasks.

We were able to achieve many other constrained tasks on the robot that are not formally described in this article due to space limits, but illustrated in the accompanying video. For example, the robot can also be constrained to move only along a vertical line in Cartesian space (i.e. equality constraint $Y = 0$ in Figure (6)), which shows the ability to satisfy tight constraints. We were also able to enforce the end-effector to remain within the horizontal plane, even under external disturbances from a human operator. In that case, as seen in the video, the robot is able to adapt automatically its configuration in order to satisfy the constraint. This confirms that arbitrary constraints can be enforced at real-time rates, without having to re-define the task (i.e. no re-tuning the cost function weights), which demonstrates the practicality and versatility of the approach.

VII. Discussion

The benchmark introduced in the unconstrained case questions what allowed this improvement. We argue that FDDP appears to be a hybrid method laying between single and multiple shooting. This idea comes both from our results on the Humanoid taichi robot, where FDDP behaves like DDP and from the theory, as it is known that once the gaps close in FDDP, they cannot re-open again. Consequently, we believe that the improvement observed in the benchmark comes from the multiple shooting formulation.

Our work follows the line of work started in the 1990s [22], [34], [36]–[39], [41] showing that standard optimization tools can be implemented specifically for OCPs by exploiting time-induced sparsity. We simply use the same tenets with SQP and ADMM as they are well-established

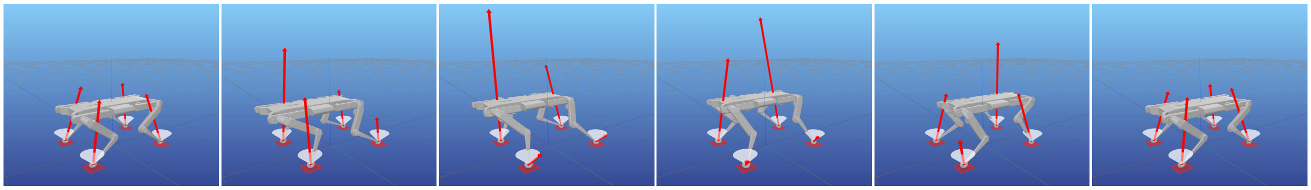


Fig. 4: Snapshots of standing motion. The red arrows represent the contact forces and the white cones are the friction constraint ($\mu = 0.8$). In the 3^{rd} and 4^{th} frames, one can see the tangential forces lying on the boundary on the friction cone.

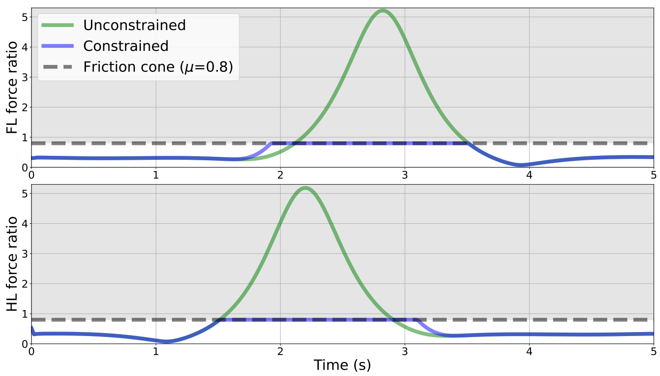


Fig. 5: Solo center-of-mass tracking task with friction cone constraints. The continuous lines represent the ratio $\frac{F_T}{F_N}$ at the front left foot, and the gray dashed line represent the friction cone constraint. Observe in the F_z plots the unconstrained OCP solution (green) crossing the friction cone constraint while the constrained OCP solution (blue) remains within the constraint.

in the optimization community. Additionally, ADMM has properties (easy to warm start and quick convergence to few iterations) that benefit MPC [63]. However, we would like to insist on the fact that the same principles could be applied to other optimization techniques. Future work would be especially interesting to modify such state-of-the-art QP solvers to exploit stagewise sparsity given that they outperformed OSQP, in general, [70], [71].

Finally, a direct by-product of the tailored implementation is the Riccati-like gains that can also enforce the additional inequality constraints. This is a natural consequence of using Riccati recursions to efficiently solve the sparse linear matrix equation that appears in the QP solver. However, it is important to note that so far, we have not used these Riccati gains on a real robot in MPC. A study on their effect on control performance and constraint satisfaction remains to be done.

During deployment, we usually early terminate the ADMM-based QP in the interest of higher re-planning frequency (100 Hz). A lower-quality solution seems to be sufficient to ensure higher reactivity and compliance on the robot. One key advantage of ADMM in this context is that the sub-QP solver can reach a reasonable solution in a few iterations and also guarantee the availability of some solution even when the constraints may be infeasible because it is based on the ADMM algorithm [63]. Both of these are desirable in non-linear MPC where obtaining a solution is essential during deployment.

VIII. Conclusion

A central message of this paper is that the existing nonlinear optimization literature already provides sufficient tools to solve OCPs with and without constraints in real time and thereby achieve state-of-the-art nonlinear MPC on real robots. We substantiated this message through a tailored, sparsity-exploiting SQP formulation that provided a unifying framework clarifying the connections between existing solvers from the robotics literature and a practical approach to achieving state-of-the-art MPC. We demonstrated through various benchmarks and hardware experiments that such a tailored SQP implementation outperformed state-of-the-art solvers based on DDP on challenging unconstrained problems. We then showed that it can efficiently solve arbitrary constrained nonlinear OCPs for MPC applications. In particular, we could enforce many nonlinear constraints on a real robot in MPC.

In conclusion, it is extremely encouraging to see that standard optimization techniques can already push the limits of closed-loop MPC for torque-controlled robots. The recent advances in Quadratic Programming suggest that more improvements are possible. This opens the way toward more advanced experimental studies of nonlinear closed-loop MPC on complex systems. In light of these results, we believe that as a robotics community, it might be worth investing more time in re-implementing efficiently established algorithms developed in the control community to push state-of-the-art constrained MPC approaches for real-world robotics applications.

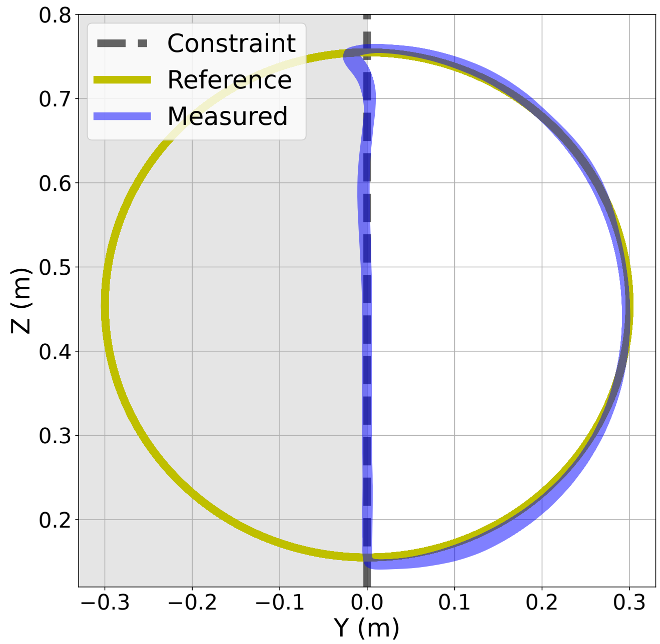
References

[1] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, "High-frequency nonlinear model predictive control of a manipulator," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7330–7336.

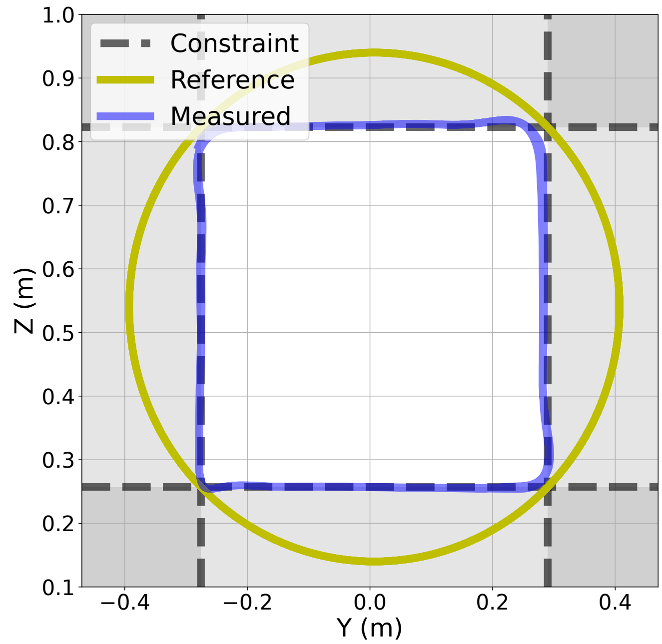
[2] A. Meduri, P. Shah, J. Viereck, M. Khadiv, I. Havoutis, and L. Righetti, "Biconmp: A nonlinear model predictive control framework for whole body motion planning," *IEEE Transactions on Robotics*, 2023.

[3] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[4] 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.



(a) The end-effector is constrained to lie within the $y > 0$ half-plane.



(b) The end-effector is constrained to lie within the intersection of 4 half-planes.

Fig. 6: End-effector position trajectories in the (y, z)-plane during the circle tracking task, subject to nonlinear constraints in Cartesian space. The gray-shaded areas represent the infeasible regions.

[5] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the hrp-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3346–3351.

[6] E. Dantec, M. Naveau, P. Fernbach, N. Villa, G. Saurel, O. Stasse, M. Taïx, 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)*. IEEE, 2022, pp. 638–644.

[7] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[8] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems." in *ICINCO (1)*. Citeseer, 2004, pp. 222–229.

[9] A. Sideris and J. E. Bobrow, "An efficient sequential linear quadratic algorithm for solving nonlinear optimal control problems," in *Proceedings of the 2005, American Control Conference, 2005*. IEEE, 2005, pp. 2275–2280.

[10] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[11] B. Plancher, Z. Manchester, and S. Kuindersma, "Constrained unscented dynamic programming," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5674–5680.

[12] M. Giftthaler, M. Neunert, M. Stäuble, J. Buchli, and M. Diehl, "A family of iterative gauss-newton shooting methods for nonlinear optimal control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[13] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7674–7679.

[14] 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 *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 2536–2542.

[15] J. Marti-Saumell, J. Solà, C. Mastalli, and A. Santamaria-Navarro, "Squash-box feasibility driven differential dynamic programming," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7637–7644.

[16] Y. Aoyama, G. Boutselis, A. Patel, and E. A. Theodorou, "Constrained differential dynamic programming revisited," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 9738–9744.

[17] C. Mastalli, W. Merkt, J. Marti-Saumell, H. Ferrolho, J. Solà, N. Mansard, and S. Vijayakumar, "A feasibility-driven approach to control-limited ddp," *Autonomous Robots*, vol. 46, no. 8, pp. 985–1005, 2022.

[18] W. Jallet, N. Mansard, and J. Carpentier, "Implicit differential dynamic programming," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1455–1461.

[19] W. Jallet, A. Bambade, N. Mansard, and J. Carpentier, "Constrained differential dynamic programming: A primal-dual augmented lagrangian approach," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 371–13 378.

[20] J. N. Nganga, H. Li, and P. Wensing, "Second-order differential dynamic programming for whole-body mpc of legged robots," in *Embracing Contacts-Workshop at ICRA 2023*, 2023.

[21] S. Wright, J. Nocedal *et al.*, "Numerical optimization," *Springer Science*, vol. 35, no. 67-68, p. 7, 1999.

[22] S. J. Wright, "Solution of discrete-time optimal control problems on parallel computers," *Parallel Computing*, vol. 16, no. 2-3, pp. 221–237, 1990.

[23] B. O'Donoghue, G. Stathopoulos, and S. Boyd, "A splitting method for optimal control," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2432–2442, 2013.

[24] D. Kouzoupis, G. Frison, A. Zanelli, and M. Diehl, "Recent advances in quadratic programming algorithms for nonlinear model predictive control," *Vietnam Journal of Mathematics*, vol. 46, no. 4, pp. 863–882, 2018.

[25] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, 2020.

[26] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "aca-

dos—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, 2022.

[27] D. Murray and S. Yakowitz, "Differential dynamic programming and newton's method for discrete optimal control problems," *Journal of Optimization Theory and Applications*, vol. 43, no. 3, pp. 395–414, 1984.

[28] K. Baumgärtner, F. Messerer, and M. Diehl, "Local convergence behaviour of generalized gauss-newton multiple shooting, single shooting and differential dynamic programming," *arXiv preprint arXiv:2301.04047*, 2023.

[29] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback mpc for torque-controlled legged robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4730–4737.

[30] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[31] L.-z. Liao and C. A. Shoemaker, "Advantages of differential dynamic programming over newton's method for discrete-time optimal control problems," Cornell University, Tech. Rep., 1992.

[32] S. Zimmermann, R. Poranne, and S. Coros, "Dynamic manipulation of deformable objects with implicit integration," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4209–4216, 2021.

[33] J. C. Dunn and D. P. Bertsekas, "Efficient dynamic programming implementations of newton's method for unconstrained optimal control problems," *Journal of Optimization Theory and Applications*, vol. 63, no. 1, pp. 23–38, 1989.

[34] J. D. O. Pantoja and D. Mayne, "Sequential quadratic programming algorithm for discrete optimal control problems with control inequality constraints," *International Journal of Control*, vol. 53, no. 4, pp. 823–836, 1991.

[35] G. Di Pillo, L. Grippo, and F. Lampariello, "A class of structured quasi-newton algorithms for optimal control problems," in *Applications of Nonlinear Programming to Optimization and Control*. Elsevier, 1984, pp. 101–107.

[36] C. Dohrmann and R. Robinett, "Efficient sequential quadratic programming implementations for equality-constrained discrete-time optimal control," *Journal of Optimization Theory and Applications*, vol. 95, pp. 323–346, 1997.

[37] S. J. Wright, "Interior point methods for optimal control of discrete time systems," *Journal of Optimization Theory and Applications*, vol. 77, no. 1, pp. 161–187, 1993.

[38] M. C. Steinbach, *A structured interior point SQP method for nonlinear optimal control problems*. Springer, 1994.

[39] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior-point methods to model predictive control," *Journal of Optimization Theory and Applications*, vol. 99, no. 3, pp. 723–757, 1998.

[40] J. Eckstein and M. C. Ferris, "Operator-splitting methods for monotone affine variational inequalities, with a parallel application to optimal control," *INFORMS Journal on Computing*, vol. 10, no. 2, pp. 218–235, 1998.

[41] C. Dohrmann and R. Robinett, "Dynamic programming method for constrained discrete-time optimal control," *Journal of Optimization Theory and Applications*, vol. 101, pp. 259–283, 1999.

[42] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit mpc," *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 18, no. 8, pp. 816–830, 2008.

[43] L. A. Rodriguez and A. Sideris, "An active set method for constrained linear quadratic optimal control," in *Proceedings of the 2010 American Control Conference*. IEEE, 2010, pp. 5197–5202.

[44] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on control systems technology*, vol. 18, no. 2, pp. 267–278, 2009.

[45] L. Vanroye, A. Sathya, J. De Schutter, and W. Decré, "Fatrop: A fast constrained optimal control problem solver for robot trajectory optimization and control," *arXiv preprint arXiv:2303.16746*, 2023.

[46] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, 2006, vol. 106, no. 1.

[47] 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.

[48] A. Romero, R. Penicka, and D. Scaramuzza, "Time-Optimal Online Replanning for Agile Quadrotor Flight," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7730–7737, 2022.

[49] A. Bratta, M. Focchi, N. Rathod, and C. Semini, "Optimization-Based Reference Generator for Nonlinear Model Predictive Control of Legged Robots," *Robotics*, vol. 12, no. 1, pp. 1–18, 2023.

[50] A. Oleinikov, S. Soltan, Z. Balgabekova, A. Bemporad, and M. Rubagotti, "Scenario-based model predictive control with probabilistic human predictions for human–robot coexistence," *Control Engineering Practice*, vol. 142, no. November 2023, p. 105769, 2024. [Online]. Available: https://doi.org/10.1016/j.conengprac.2023.105769

[51] S. Katayama, M. Murooka, and Y. Tazaki, "Model predictive control of legged and humanoid robots: models and algorithms," *Advanced Robotics*, vol. 37, no. 5, pp. 298–315, 2023.

[52] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[53] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "Osqp: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[54] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 1999.

[55] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, jun 2000.

[56] W. Lee, "Tridiagonal matrices: Thomas algorithm," *MS6021, Scientific Computation, University of Limerick*, 2011.

[57] M. Diehl, "Lecture notes on optimal control and estimation," *Lecture Notes on Optimal Control and Estimation*, 2014.

[58] S. J. Wright, "Partitioned dynamic programming for optimal control," *SIAM Journal on optimization*, vol. 1, no. 4, pp. 620–642, 1991.

[59] J. V. Frasch, S. Sager, and M. Diehl, "A parallel quadratic programming method for dynamic optimization problems," *Mathematical programming computation*, vol. 7, pp. 289–329, 2015.

[60] B. L. Nicholson, W. Wan, S. Kameswaran, and L. T. Biegler, "Parallel cyclic reduction strategies for linear systems that arise in dynamic optimization problems," *Computational Optimization and Applications*, vol. 70, pp. 321–350, 2018.

[61] S. Särkkä and Á. F. García-Fernández, "Temporal parallelization of dynamic programming and linear quadratic control," *IEEE Transactions on Automatic Control*, vol. 68, no. 2, pp. 851–866, 2022.

[62] S. Caron, A. Zaki, P. Otta, D. Arnström, and J. Carpentier, "qpsolver benchmark: Benchmark for quadratic programming solvers available in Python," Sep. 2023. [Online]. Available: https://github.com/qpsolvers/qpsolvers_benchmark

[63] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[64] G. Frison, "General rights Algorithms and Methods for High-Performance Model Predictive Control," 2015. [Online]. Available: www.compute.dtu.dk

[65] R. Fletcher and S. Leyffer, "Nonlinear programming without a penalty function," *Mathematical programming*, vol. 91, pp. 239–269, 2002.

[66] M. Diehl, H. G. Bock, H. Diedam, P. B. Wieber, P.-b. W. Fast, and D. Multiple, "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control," p. 28, 2009.

[67] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," *Proceedings of the 2019 IEEE/SICE International Symposium on System Integration, SII 2019*, pp. 614–619, 2019.

[68] J. Carpentier and N. Mansard, "Analytical derivatives of rigid body dynamics algorithms," in *Robotics: Science and systems (RSS 2018)*, 2018.

[69] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Spröwitz, and L. Righetti, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[70] A. Bambade, S. El-Kazdadi, A. Taylor, and J. Carpentier, "Proxqp: Yet another quadratic programming solver for robotics and beyond," in *RSS 2022-Robotics: Science and Systems*, 2022.

[71] R. Schwan, Y. Jiang, D. Kuhn, and C. N. Jones, "Piqp: A proximal interior-point quadratic programming solver," *arXiv preprint arXiv:2304.00290*, 2023.