Parallel and Proximal Constrained Linear-Quadratic Methods for Real-Time Nonlinear MPC

Wilson Jallet*† Ewen Dantec† Etienne Arlaud† Nicolas Mansard*‡ Justin Carpentier†
*LAAS-CNRS, University of Toulouse
31400 Toulouse, France
{wjallet,nmansard}@laas.fr
†Inria - Département d'Informatique de l'École Normale Supérieure, PSL Research University
75012 Paris, France
{wilson.jallet,ewen.dantec,etienne.arlaud,justin.carpentier}@inria.fr
‡ANITI, University of Toulouse
Toulouse, France

***Abstract*—Recent strides in nonlinear model predictive control (NMPC) underscore a dependence on numerical advancements to efficiently and accurately solve large-scale problems. Given the substantial number of variables characterizing typical whole-body optimal control (OC) problems —often numbering in the thousands— exploiting the sparse structure of the numerical problem becomes crucial to meet computational demands, typically in the range of a few milliseconds. Addressing the linear-quadratic regulator (LQR) problem is a fundamental building block for computing Newton or Sequential Quadratic Programming (SQP) steps in direct optimal control methods. This paper concentrates on equality-constrained problems featuring implicit system dynamics and dual regularization, a characteristic of advanced interior-point or augmented Lagrangian solvers. Here, we introduce a parallel algorithm for solving an LQR problem with dual regularization. Leveraging a rewriting of the LQR recursion through block elimination, we first enhanced the efficiency of the serial algorithm and then subsequently generalized it to handle parametric problems. This extension enables us to split decision variables and solve multiple subproblems concurrently. Our algorithm is implemented in our nonlinear numerical optimal control library ALIGATOR[1]. It showcases improved performance over previous serial formulations and we validate its efficacy by deploying it in the model predictive control of a real quadruped robot.**

I. Introduction

In this paper, we introduce a parallel algorithm to enhance the efficiency of model-predictive control (MPC) solvers [49, 16]. The computational complexity of these solvers is a pivotal factor in numerical optimal control – in particular, to allow their implementation on real hardware. More specifically, we consider linear-quadratic (LQ) problems (i.e., with quadratic cost and linear constraints), which are a fundamental block of such iterative solvers. LQ is the standard form of subproblems in many direct methods derived from Newton's method [20] such as sequential quadratic programming (SQP) [19, 24], differential dynamic programming (DDP), and iterative LQR [27, 49].

In particular, the classical linear-quadratic regulator (LQR) is an LQ problem defined in terms of explicit linear dynamics and an unconstrained objective. The most well-known method for solving the classical LQR is the Riccati recursion, which can be derived by dynamic programming [4]. Yet, direct equivalence with block-factorization methods has been drawn [51] and recently exploited in robotics [31].

Over the past decade, proposals have been given for the resolution of nonlinear equality-constrained problems. Most solutions essentially extend the Riccati recursion approaches to properly account for equality constraints, by exploiting projection or nullspace approaches [23, 33, 50] or augmented Lagrangian-based approaches [32, 26]. While solving the LQR is often a bottleneck in recent efficient optimal control solvers [21, 36, 22], most of them rely on sequential implementation without exploiting the parallelization capabilities of modern processing units.

Several methods were previously developed for the parallel solving of LQ problems: [18] is based on a Gauss-Seidel modification of the Riccati backward sweep, ADMM schemes separating costs and constraints [48] and conjugate-gradient methods [1] (efficient on GPUs), all with intrinsic approximate (iterative) convergence at linear rate.

On the other hand, Wright [51, 52] looked at direct methods for solving linear-quadratic problems on parallel architectures. (first with a tailored banded matrix LU solver [51], then [52] with a specialized method using dynamic programming after partitioning the problem, with an extension to active-set methods). More recently, Nielsen and Axehill [38] subdivide the LQR problem into subproblems with state-control linkage constraints; this approach involves computing nullspace matrices to handle infeasible subproblems. Then, [39] introduces a variant based on parameterizing each subproblem on the next subproblem value function's parameters. Laine and Tomlin [34] suggest subdivision of the LQR problem into a set of subproblems with state linkage constraints at the endpoints; the linkage constraints' multipliers satisfy a global system of equations which is solved by least-squares.

In this paper, we propose a general direct solver for LQ problems with implicit dynamics and additional equality constraints, leveraging parameterization to formulate a parallel algorithm, a similar idea to Nielsen and Axehill [39], Laine and Tomlin

[1]Repository: https://github.com/Simple-Robotics/aligator/

[34]. This novel algorithm is implemented in C++, and used as a backend in a nonlinear trajectory optimizer which handles both equality and inequality constraints using an augmented Lagrangian method. Its effectiveness is demonstrated on various robotic benchmarks and by implementing an MPC scheme on a real quadruped robot. This paper follows up from our prior work on augmented Lagrangian methods for numerical optimal control with implicit dynamics and constraints [29, 28].

After recalling the equality-constrained LQR problem in Section II, we first derive serial Riccati equations in the proximal setting in Section III, for which we build a block-sparse factorization in Section III-C. This formulation is extended in Section IV to parametric LQ problems, which we finally use in Section V to build a parallel algorithm and discuss it with respect to the literature. The application of our algorithm in proximal nonlinear trajectory optimization is described in Section VI, along with performance benchmarks and experiments in Section VIII.

Notation

In this paper, we denote, $\mathbb{R}^{n\times m}$ the set of $n\times m$ real matrices, and $\mathbb{S}_n(\mathbb{R})$ the set of $n\times n$ symmetric real matrices. We denote $[\![a,b]\!] = \{a,a+1,\dots,b\}$ the interval of integers between two integers $a \leqslant b$. We will use italic bold letters to denote tuples of vectors $\boldsymbol{z} = (z_0,\dots,z_k)$ of possibly varying dimensions, and given an index set $\mathcal{I} \subseteq [\![0,k]\!]$, we denote by $\boldsymbol{z}_{\mathcal{I}}$ the subset $(z_i)_{i\in\mathcal{I}}$ (replacing $\mathcal{I}$ by its intersection with $[\![0,k]\!]$ when not contained in the former, by abuse of notation). For linear systems $Hz+g=0$, we will use the shorthand notation

$$\overset{z}{[H \mid g]}$$

for compactness.

II. Equality-constrained linear-quadratic Problems

In this section, we will recall the equality-constrained linear-quadratic (LQ) problem, its optimality conditions, and proximal methods to solving it.

A. Problem statement

We consider the following equality-constrained LQ problem:

$$\min_{\boldsymbol{x},\boldsymbol{u}} J(\boldsymbol{x},\boldsymbol{u}) \overset{\text{def}}{=} \sum_{t=0}^{N-1} \ell_t(x_t,u_t) + \ell_N(x_N) \tag{1a}$$
$$\text{s.t. } A_t x_t + B_t u_t + E_t x_{t+1} + f_t = 0 \tag{1b}$$
$$C_t x_t + D_t u_t + h_t = 0,\ t=0,\dots,N-1 \tag{1c}$$
$$C_N x_N + h_N = 0 \tag{1d}$$
$$G_0 x_0 + g_0 = 0 \tag{1e}$$

with the quadratic running and terminal cost functions

$$\ell_t(x_t,u_t) = \frac{1}{2}\begin{bmatrix}x_t\\u_t\end{bmatrix}^\top\begin{bmatrix}Q_t & S_t\\ S_t^\top & R_t\end{bmatrix}\begin{bmatrix}x_t\\u_t\end{bmatrix} + q_t^\top x_t + r_t^\top u_t, \tag{2a}$$
$$\ell_N(x_N) = \frac{1}{2}x_N^\top Q_N x_N + q_N^\top x_N \tag{2b}$$

for a discrete dynamics along the time horizon $t=0,\dots,N$, with state vector $x_t \in \mathbb{R}^{n_x}$, control input $u_t \in \mathbb{R}^{n_u}$. The parameters of the problem are:

- dynamics matrices $A_t, E_t \in \mathbb{R}^{n_x\times n_x}$, $B_t \in \mathbb{R}^{n_x\times n_u}$, and $f_t \in \mathbb{R}^{n_x}$,
- constraint matrices $C_t \in \mathbb{R}^{n_c^t\times n_x}$, $D_t \in \mathbb{R}^{n_c^t\times n_u}$, $h_t \in \mathbb{R}^{n_c^t}$ ($n_c^t \in \mathbb{N}$ being the number of rows),
- cost matrices $Q_t \in \mathbb{S}_{n_x}(\mathbb{R})$, $R_t \in \mathbb{S}_{n_u}(\mathbb{R})$, $S_t \in \mathbb{R}^{n_x\times n_u}$, $q_t \in \mathbb{R}^{n_x}$ and $r_t \in \mathbb{R}^{n_u}$, and
- initial constraint matrix $G_0 \in \mathbb{R}^{n_g\times n_x}$ and vector $g_0 \in \mathbb{R}^{n_g}$.

In most cases, the initial constraint would be $x_0 - \bar{x} = 0$ ($G_0 = -I$, $g_0 = \bar{x} \in \mathbb{R}^{n_x}$).

Remark 1. *We consider the general case of implicit dynamics* (1b)*, with the particular case of explicit dynamics obtained by $E_t = -I$.*

Remark 2. *The LQ problem* (1) *is not always feasible.*

B. Lagrangian and KKT conditions

We introduce the following Hamiltonian function:

$$\begin{aligned} H_t(x,u,\nu,\lambda) \overset{\text{def}}{=} \ell_t(x,u) &+ \lambda^\top(A_t x + B_t u + f_t) \\ &+ \nu^\top(C_t x + D_t u + h_t) \end{aligned} \tag{3}$$

for $t=0,\dots,N-1$, and terminal stage Lagrangian

$$\mathscr{L}_N(x,\nu) \overset{\text{def}}{=} \ell_N(x) + \nu^\top(C_N x + h_N)$$

Using this notation, the Lagrangian of Problem (1) reads

$$\begin{aligned}\mathscr{L}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\nu},\boldsymbol{\lambda}) &\overset{\text{def}}{=} \lambda_0^\top(G_0x_0+g_0) \\ &+ \sum_{t=0}^{N-1} H_t(x_t,u_t,\nu_t,\lambda_{t+1}) + \lambda_{t+1}^\top E_t x_{t+1} + \mathscr{L}_N(x_N,\nu_N).\end{aligned} \tag{4}$$

Problem (1) has linear constraints, hence linear constraint qualifications apply and the Karush-Kuhn-Tucker (KKT) optimality conditions read (with, for convenience, $E_{-1} = G_0$):

$$-E_{t-1}^\top\lambda_t = Q_t x_t + S_t u_t + A_t^\top\lambda_{t+1} + C_t^\top\nu_t + q_t \tag{5a}$$
$$0 = S_t^\top x_t + R_t u_t + B_t^\top\lambda_{t+1} + D_t^\top\nu_t + r_t \tag{5b}$$
$$0 = C_t x_t + D_t u_t + h_t \tag{5c}$$
$$0 = A_t x_t + B_t u_t + E_t x_{t+1} + f_t \tag{5d}$$

for $0 \leqslant t \leqslant N-1$ with terminal and initial conditions:

$$-E_{N-1}^\top\lambda_N = Q_N x_N + C_N^\top\nu_N + q_N \tag{5e}$$
$$0 = C_N x_N + h_N \tag{5f}$$
$$0 = G_0 x_0 + g_0. \tag{5g}$$

C. Proximal regularization of the LQ problem

In this subsection, we introduce a proximal regularization of the LQ problem (1) in its dual variables and derive the corresponding optimality conditions. Leveraging proximal regularization in constrained optimization settings is a generic way to tackle ill-posed problems (e.g., rank deficient constraints) in the optimization literature [40, 6]. The saddle-point formulation of (1) is

$$\min_{\boldsymbol{x},\boldsymbol{u}} \max_{\boldsymbol{\lambda},\boldsymbol{\nu}} \mathscr{L}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\nu},\boldsymbol{\lambda}).$$

The corresponding dual proximal-point iteration from previous estimates $(\boldsymbol{\lambda}^e, \boldsymbol{\nu}^e)$ of the co-state and path multipliers is:

$$\min_{\boldsymbol{x},\boldsymbol{u}} \max_{\boldsymbol{\lambda},\boldsymbol{\nu}} \mathscr{L}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{\nu},\boldsymbol{\lambda}) - \frac{\mu}{2} \left\| [\boldsymbol{\lambda},\boldsymbol{\nu}] - [\boldsymbol{\lambda}^e,\boldsymbol{\nu}^e] \right\|_2^2, \tag{6}$$

where $\mu > 0$ is the proximal parameter. This *proximal LQ problem* is known [43] to be equivalent to finding a minimizer of the augmented Lagrangian associated with (1). We denote by $\bar{g}_0$, $\bar{f}_t$ and $\bar{h}_t$ the “shifted” right-hand side quantities

$$\bar{g}_0 \overset{\text{def}}{=} g_0 + \mu\lambda_0^e,\ \bar{f}_t \overset{\text{def}}{=} f_t + \mu\lambda_{t+1}^e,\ \bar{h}_t \overset{\text{def}}{=} h_t + \mu\nu_t^e. \tag{7}$$

Then, the KKT conditions of the proximal LQ problem are given by (5a), (5b) and

$$0 = A_t x_t + B_t u_t + E_t x_{t+1} + \bar{f}_t - \mu\lambda_{t+1} \tag{8a}$$
$$0 = C_t x_t + D_t u_t + \bar{h}_t - \mu\nu_t \tag{8b}$$
$$0 = C_N x_N + \bar{h}_N - \mu\nu_N \tag{8c}$$
$$0 = G_0 x_0 + \bar{g}_0 - \mu\lambda_0, \tag{8d}$$

This system of equations arises when trying to solve (1) through a proximal iteration scheme, but it also arises when applying an augmented Lagrangian method to a general nonlinear control problem as in section VI.

A Riccati-like recursion has been proposed in [29, 28] to solve the proximal LQ problem. Yet, it requires solving at each stage of the recursion a larger linear system than the classical (unconstrained, unregularized) Riccati setting.

III. Riccati Equations for the proximal LQ Problem

In this section, we will present a generalization of the Riccati recursion, initially introduced by [29], which is akin to taking successive Schur complements. This requires solving large symmetric linear systems at each stage of the recursion, for which we will provide an efficient, structure-exploiting approach in section III-C. A self-contained refresher of the classical Riccati recursion is given in appendix A.

A. Solving the proximal LQ by block substitution

The KKT conditions can be rewritten as a block-banded linear system of the form:

$$\begin{array}{cccccccccccc|c}
x_0 & u_0 & \nu_0 & \lambda_1 & x_1 & u_1 & \nu_1 & \lambda_2 & \dots & \lambda_N & x_N & \nu_N & \\
Q_0 & S_0 & C_0^\top & A_0^\top & & & & & & & & & q_0 \\
S_0^\top & R_0 & D_0^\top & B_0^\top & & & & & & & & & r_0 \\
C_0 & D_0 & -\mu I & & & & & & & & & & \bar{h}_0 \\
A_0 & B_0 & & -\mu I & E_0 & & & & & & & & \bar{f}_0 \\
 & & & E_0^\top & Q_1 & S_1 & C_1^\top & A_1^\top & & & & & q_1 \\
 & & & & S_1^\top & R_1 & D_1^\top & B_1^\top & & & & & r_1 \\
 & & & & C_1 & D_1 & -\mu I & & & & & & \bar{h}_1 \\
 & & & & A_1 & B_1 & & -\mu I & E_1 & \ddots & & & \bar{f}_1 \\
 & & & & & & & E_1^\top & \ddots & & & & \vdots \\
 & & & & & & & & \ddots & & E_{N-1} & & \bar{f}_{N-1} \\
 & & & & & & & & & E_{N-1}^\top & Q_N & C_N^\top & q_N \\
 & & & & & & & & & & C_N & -\mu I & \bar{h}_N
\end{array} \tag{9}$$

For simplicity of presentation, we will consider, in this section, the case where $N = 2$.

1) Terminal stage: Starting from the lower-right block in the unknowns (x_2, ν_2), we can express the terminal system in the unknowns (x_2, ν_2),

$$\begin{array}{cc|c}
x_2 & \nu_2 & \\
Q_2 & C_2^\top & q_2 + E_1^\top \lambda_2 \\
C_2 & -\mu I & \bar{h}_2
\end{array}. \tag{10}$$

A Schur complement in x_2 leads to $\nu_2 = \frac{1}{\mu}(\bar{h}_2 + C_2 x_2)$ and the following equation:

$$P_2 x_2 + p_2 + E_1^\top \lambda_2 = 0 \tag{11}$$

where $P_2 = Q_2 + \frac{1}{\mu} C_2^\top C_2$ and $p_2 = q_2 + \frac{1}{\mu} C_2^\top \bar{h}_2$ correspond to the terminal cost-to-go matrix and vector in the classical Riccati recursion.[1]

2) Middle blocks: Equations (9) and (11) lead to the following system in $(x_1, u_1, \nu_1, \lambda_2, x_2)$ (where the unknown λ_1 is a parameter):

$$\begin{array}{ccccc|c}
x_1 & u_1 & \nu_1 & \lambda_2 & x_2 & \\
Q_1 & S_1 & C_1^\top & A_1^\top & & q_1 + E_0^\top \lambda_1 \\
S_1^\top & R_1 & D_1^\top & B_1^\top & & r_1 \\
C_1 & D_1 & -\mu I & & & \bar{h}_1 \\
A_1 & B_1 & & -\mu I & E_1 & \bar{f}_1 \\
 & & & E_1^\top & P_2 & p_2
\end{array}. \tag{12}$$

As x_1 is unknown, we will solve this equation parametrically. We introduce the primal-dual feedforward (resp. feedback) gains $(k_1, \zeta_1, \omega_2, a_1)$ (resp. $(K_1, Z_1, \Omega_2, M_1)$) for the control, constraint multiplier, co-state and next state, so that the parametric solution in $(u_1, \nu_1, \lambda_2, x_2)$ is:

$$\begin{aligned} u_1 &= k_1 + K_1 x_1, & \nu_1 &= \zeta_1 + Z_1 x_1 \\ \lambda_2 &= \omega_2 + \Omega_2 x_1 & x_2 &= a_1 + M_1 x_1. \end{aligned} \tag{13}$$

These primal-dual gains satisfy the linear system:

$$\begin{bmatrix} R_1 & D_1^\top & B_1^\top & \\ D_1 & -\mu I & & \\ B_1 & & -\mu I & E_1 \\ & & E_1^\top & P_2 \end{bmatrix} \begin{bmatrix} k_1 & K_1 \\ \zeta_1 & Z_1 \\ \omega_2 & \Omega_2 \\ a_1 & M_1 \end{bmatrix} = - \begin{bmatrix} r_1 & S_1^\top \\ \bar{h}_1 & C_1 \\ \bar{f}_1 & A_1 \\ p_1 & 0 \end{bmatrix}. \tag{14}$$

Once the matrix-matrix system (14) has been solved, we can substitute the expressions of $(u_1, \nu_1, \lambda_2, x_2)$ as functions of x_1 into the first line of (12), which reduces to

$$E_0^\top \lambda_1 + P_1 x_1 + p_1 = 0 \tag{15}$$

where we have introduced the new cost-to-go matrix and vector for stage $t = 1$:

$$P_1 = Q_1 + S_1 K_1 + C_1^\top Z_1 + A_1^\top \Omega_2 \tag{16a}$$
$$p_1 = q_1 + S_1 k_1 + C_1^\top \zeta_1 + A_1^\top \omega_2. \tag{16b}$$

[1]This can be seen as follows: if $E_1 = -I$ and there are no constraints, then $P_2 = Q_2$ and $p_2 = q_2$, then (11) reduces to $\lambda_2 = Q_2 x_2 + q_2$ which is the terminal condition in the usual Riccati recursion, see appendix A.

This equation has the expected form to close out the recursion. Indeed, the remaining system has the form

$$\begin{array}{c} \begin{array}{cccccc} \lambda_0 & x_0 & u_0 & \nu_0 & \lambda_1 & x_1 \end{array} \\ \left[\begin{array}{cccccc|c} -\mu I & G_0 & & & & & \bar{g}_0 \\ G_0^\top & Q_0 & S_0 & C_0^\top & A_0^\top & & q_0 \\ & S_0^\top & R_0 & D_0^\top & B_0^\top & & r_0 \\ & C_0 & D_0 & -\mu I & & & \bar{h}_0 \\ & A_0 & B_0 & & -\mu I & E_0 & \bar{f}_0 \\ & & & & E_0^\top & P_1 & p_1 \end{array}\right] \end{array} \tag{17}$$

which has the same structure as the initial problem. We can iterate the previous derivation to obtain $(u_0, \nu_0, \lambda_1, x_1)$ as affine functions of x_0, then a cost-to-go matrix and vector (P_0, p_0).

3) Initial stage: If x_0 is fixed (e.g. in classical DDP algorithms [37, 49]) then we are done. However, if x_0 is not a fixed variable and is indeed a decision variable (e.g. in direct multiple-shooting schemes [19]) along with λ_0, then they together satisfy the symmetric system:

$$\begin{array}{c} \begin{array}{cc} x_0 & \lambda_0 \end{array} \\ \left[\begin{array}{cc|c} P_0 & G_0^\top & p_0 \\ G_0 & -\mu I & \bar{g}_0 \end{array}\right] \end{array}. \tag{18}$$

B. Riccati-like algorithm

The overall Algorithm 1 is finally obtained (for any $N \geq 2$) by repeating Step 2) (the middle blocks) of section III-A from time step $t = N-1$ down to $t = 0$, and handling the initial stage as previously described in Step 3). Although obtained differently, it is the same as proposed in [28].

At each stage of the recursion, the solution of system (14) can be obtained by dense matrix decomposition (e.g. LU, $LDL^\top$, or Bunch-Kaufman [9]), which carries a time complexity of order $\mathcal{O}((n_u + n_c + 2n_x)^3)$. However, if appropriate assumptions are made, a dense factorization routine can be avoided and replaced by one which exploits the block-sparsity of (12) as we will outline in the next section III-C.

C. Block-sparse factorization for the stage KKT system

Rather than applying a dense factorization procedure, we propose to leverage the block-sparse structure of (12)-(14) to build a more efficient algorithm than the one proposed in [28]. It avoids solving a system of size $n_u + n_c + 2n_x$ by a well-chosen substitution using the assumption that E_t is invertible (with $E_t = -I$ when the dynamics are explicit).

This assumption can be motivated as follows: discretization of an ODE, as we mention in section VI, can lead to implicit discrete dynamics $\phi_t(x_t, u_t, x_{t+1}) = 0$ (this is the case with e.g. Runge-Kutta methods and variational integrators [29]). Provided a small enough timestep, the solution to this implicit equation is unique: in fact there is[2] a differentiable, local inverse map F defined on a neighbourhood $\mathcal{O}$ of (x_t, u_t) such that $\phi_t(x, u, F(x, u)) = 0$ for $(x, u) \in \mathcal{O}$.

A possible alternative would be to assume that the "cost-to-go" matrix P_2 in (12) is nonsingular, and compute its inverse to perform a Schur complement. The assumption can be satisfied

[2]This result stems from the Implicit Function Theorem.

Algorithm 1: Generalized Riccati equations for proximal, constrained LQ problem

```
   Data: Cost and constraint matrices
         $Q_t, S_t, R_t, q_t, r_t, A_t, B_t, C_t, E_t, D_t, f_t, h_t$
 1 $P_N \leftarrow Q_N + \frac{1}{\mu} C_N^\top C_N$;
 2 $p_N \leftarrow q_N + \frac{1}{\mu} C_N^\top \bar{h}_N$;
   // Backward pass
 3 for $t = N-1$ to $t = 0$ do
 4     $[k_t, \zeta_t, \omega_{t+1}, a_t, K_t, Z_t, \Omega_{t+1}, M_t] \leftarrow$ solve (14);
       // Set cost-to-go following (16)
 5     $P_t \leftarrow Q_t + S_t K_t + C_t^\top Z_t + A_t^\top \Omega_{t+1}$;
 6     $p_t \leftarrow q_t + S_t k_t + C_t^\top \zeta_t + A_t^\top \omega_{t+1}$;
 7 $(x_0, \lambda_0) \leftarrow$ solve (18) ;    // or impose value of $x_0$
   // Forward pass
 8 for $t = 0$ to $N-1$ do
 9     $u_t \leftarrow k_t + K_t x_t$;
10     $\nu_t \leftarrow \zeta_t + Z_t x_t$;
11     $\lambda_{t+1} \leftarrow \omega_{t+1} + \Omega_{t+1} x_t$ ;
12     $x_{t+1} \leftarrow a_t + M_t$;
13 $\nu_N = \frac{1}{\mu}(\bar{h}_N + C_N x_N)$;
```

by requiring that the pure-state cost matrices Q_t be definite positive. This is a drawback for problems with e.g. semidefinite terminal cost Hessians (for instance, when only the joint velocities are penalized but not the joint configurations).

The mathematical derivation for the block-sparse factorization is postponed to appendix E as its understanding is not necessary to continue on to the next section. We present this as a secondary contribution of this paper, which we have implemented and evaluated in the experimental section.

Discussion

In conclusion, this section reformulated the backward recursion for the proximal LQ problem introduced in [28] and allowed us to suggest a more efficient alternative. More importantly, this formulation paves the road towards our parallel algorithm. Next section will develop a variant of algorithm 4 geared toward parametric problems, which will be the cornerstone to design our parallel formulation.

IV. Extension to Parametric LQ problems

In this section, we consider problems for which the Lagrangian has an additional affine term with parameter vector $\theta \in \mathbb{R}^{n_\theta}$. Differentiable, parametric optimal control in general has been covered in recent literature [2, 16, 41, 5]. In particular, the case of constrained problems with proximal regularization has been covered by Bounou et al. [5], where (12). In this subsection, we extend the block-sparse approach we presented in section III-C to parametric problems. We will build upon this in the next section to derive a method for parallel resolution of LQ problems (1).

A. *Parametric Lagrangians*

In this section, we consider a parametric LQ problem with a Lagrangian of the form (without initial condition):

$$\begin{aligned}\mathscr{L}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda};\theta) &= \bar{\mathscr{L}}_N(x_N,v_N;\theta)\\ &+\sum_{t=0}^{N-1}\bar{H}_t(x_t,u_t,v_t,\lambda_{t+1};\theta)+\lambda_{t+1}^\top E_t x_{t+1}\end{aligned} \tag{19}$$

where each parametric Hamiltonian $\bar{H}_t$ reads

$$\begin{aligned}&\bar{H}_t(x,u,v,\lambda;\theta) =\\ &\qquad H_t(x,u,v,\lambda)+\theta^\top(\Phi_t^\top x+\Psi_t^\top u+\gamma_t)+\tfrac{1}{2}\theta^\top\Gamma_t\theta\end{aligned} \tag{20}$$

where H_t contains the non-parametric terms of the Hamiltonian, $\gamma_t\in\mathbb{R}^{n_\theta}$ and $\Phi_t\in\mathbb{R}^{n_x\times n_\theta}$, $\Psi_t\in\mathbb{R}^{n_u\times n_\theta}$, $\Gamma_t\in\mathbb{R}^{n_\theta\times n_\theta}$. Similarly, the parametric terminal Lagrangian reads

$$\bar{\mathscr{L}}_N(x,v;\theta)=\mathscr{L}_N(x,v)+\theta^\top(\Phi_N^\top x+\gamma_N)+\frac{1}{2}\theta^\top\Gamma_N\theta. \tag{21}$$

This representation can stem from having additional affine cost terms in the LQ problem. We could seek to compute the sensitivities of either the optimal value or the optimal primal-dual trajectory $\boldsymbol{\xi}^\star=(\boldsymbol{x}^\star,\boldsymbol{u}^\star,\boldsymbol{v}^\star,\boldsymbol{\lambda}^\star)$ with respect to these additional terms [2].

We denote by $\mathcal{E}$ the value function of problem (1) under parameters θ, defined in saddle-point form by

$$\mathcal{E}(x_0,\theta)\overset{\text{def}}{=}\min_{\boldsymbol{x},\boldsymbol{u}}\max_{\boldsymbol{v},\boldsymbol{\lambda}}\mathscr{L}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda};\theta) \tag{22}$$

Similarly, we denote by $\mathcal{E}_\mu(x_0,\theta)$ the value function of the dual proximal-regularization (6) (as in section II) for $\mu>0$:

$$\mathcal{E}_\mu(x_0,\theta)\overset{\text{def}}{=}\min_{\boldsymbol{x},\boldsymbol{u}}\max_{\boldsymbol{v},\boldsymbol{\lambda}}\mathscr{L}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda};\theta)-\frac{\mu}{2}\|[\boldsymbol{\lambda},\boldsymbol{v}]-[\boldsymbol{\lambda}_e,\boldsymbol{v}_e]\|^2. \tag{23}$$

B. *Expression of the value function $\mathcal{E}_\mu(x_0,\theta)$*

Denote $\boldsymbol{\xi}=(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda})$ the collection of primal-dual variables. We have the following property, derived from the Schur complement lemma (see appendix B):

Proposition 3. *$\mathcal{E}_\mu$ is a quadratic function in (x_0,θ). There exist $\sigma_0\in\mathbb{R}^{n_\theta}$, $\Lambda_0\in\mathbb{R}^{n_x\times n_\theta}$ and $\Sigma_0\in\mathbb{R}^{n_\theta\times n_\theta}$ such that*

$$\mathcal{E}_\mu(x_0,\theta)=\frac{1}{2}\begin{bmatrix}x_0\\ \theta\end{bmatrix}^\top\begin{bmatrix}P_0 & \Lambda_0\\ \Lambda_0^\top & \Sigma_0\end{bmatrix}\begin{bmatrix}x_0\\ \theta\end{bmatrix}+p_0^\top x_0+\sigma_0^\top\theta. \tag{24}$$

Proof: The min-maximand in (23) is a quadratic function in $(\boldsymbol{\xi},\theta)$ which is convex-concave in $\boldsymbol{\xi}$. Then, we apply the lemma in appendix B2 to (23), choosing (x_0,θ) as the parameter (z in the lemma) in the saddle-point. ■

Furthermore, the lemma in appendix B2 yields

$$\begin{aligned}\sigma_0 &= \mathscr{L}_\theta-\mathscr{L}_{\theta\boldsymbol{\xi}}\mathscr{L}_{\boldsymbol{\xi}\boldsymbol{\xi}}^{-1}\mathscr{L}_{\boldsymbol{\xi}\theta}\\ \Lambda_0 &= \mathscr{L}_{x_0\theta}-\mathscr{L}_{x_0\boldsymbol{\xi}}\mathscr{L}_{\boldsymbol{\xi}\boldsymbol{\xi}}^{-1}\mathscr{L}_{\boldsymbol{\xi}\theta}\\ \Sigma_0 &= \mathscr{L}_{\theta\theta}-\mathscr{L}_{\theta\boldsymbol{\xi}}\mathscr{L}_{\boldsymbol{\xi}\boldsymbol{\xi}}^{-1}\mathscr{L}_{\boldsymbol{\xi}\theta},\end{aligned} \tag{25}$$

where the following derivatives in θ are simple:

$$\mathscr{L}_\theta=\sum_{t=0}^N\gamma_t,\ \mathscr{L}_{\theta\theta}=\sum_{t=0}^N\Gamma_t,\ \mathscr{L}_{x_0\theta}=\Phi_0. \tag{26}$$

Solving $\boldsymbol{\xi}^0=-\mathscr{L}_{\boldsymbol{\xi}\boldsymbol{\xi}}^{-1}\mathscr{L}_{\boldsymbol{\xi}}$ is handled by the variant of the Riccati recursion introduced in the previous section III.

Denote by $\partial_\theta\boldsymbol{\xi}=(\partial_\theta\boldsymbol{x},\partial_\theta\boldsymbol{u},\partial_\theta\boldsymbol{v},\partial_\theta\boldsymbol{\lambda})=-\mathscr{L}_{\boldsymbol{\xi}\boldsymbol{\xi}}^{-1}\mathscr{L}_{\boldsymbol{\xi}\theta}$ the sensitivity matrix, such that the solution of (23) is

$$\boldsymbol{\xi}^\star(\theta)=\boldsymbol{\xi}^0+\partial_\theta\boldsymbol{\xi}\cdot\theta.$$

Solving for $\partial_\theta\boldsymbol{\xi}$ requires a slight alteration of our generalized Riccati recursion.

The terminal stage is

$$\begin{array}{cc|c}\partial_\theta x_N & \partial_\theta v_N & \\ \hline Q_N & C_N^\top & \Phi_N+E_{N-1}^\top\partial_\theta\lambda_N\\ C_N & -\mu I & 0\end{array}. \tag{27}$$

Defining $\Lambda_N\overset{\text{def}}{=}\Phi_N\in\mathbb{R}^{n_x\times n_\theta}$, this leads to

$$P_N\partial_\theta x_N+\Lambda_N+E_{N-1}^\top\partial_\theta\lambda_N=0.$$

For non-terminal stages, we obtain a system of equations

$$\begin{array}{ccccc|c}\partial_\theta x_t & \partial_\theta u_t & \partial_\theta v_t & \partial_\theta\lambda_{t+1} & \partial_\theta x_{t+1} & \\ \hline Q_t & S_t & C_t^\top & A_t^\top & & \Phi_t+E_{t-1}^\top\partial_\theta\lambda_t\\ S_t^\top & R_t & D_t^\top & B_t^\top & & \Psi_t\\ C_t & D_t & -\mu I & & & 0\\ A_t & B_t & & -\mu I & E_t & 0\\ & & & E_t^\top & P_{t+1} & \Lambda_{t+1}\end{array} \tag{28}$$

which can be solved as before, by introducing a matrix right-hand side variant of (14):

$$\begin{array}{cccc|c}K_t^\theta & Z_t^\theta & \Omega_{t+1}^\theta & M_{t+1}^\theta & \\ \hline R_t & D_t^\top & B_t^\top & & \Psi_t\\ D_t & -\mu I & & & 0\\ B_t & & -\mu I & E_t & 0\\ & & E_t^\top & P_{t+1} & \Lambda_{t+1}\end{array} \tag{29}$$

such that

$$\begin{bmatrix}\partial_\theta u_t\\ \partial_\theta v_t\\ \partial_\theta\lambda_{t+1}\\ \partial_\theta x_{t+1}\end{bmatrix}=\begin{bmatrix}K_t\partial_\theta x_t+K_t^\theta\\ Z_t\partial_\theta x_t+Z_t^\theta\\ \Omega_{t+1}\partial_\theta x_t+\Omega_{t+1}^\theta\\ M_t\partial_\theta x_t+M_t^\theta\end{bmatrix}. \tag{30}$$

This leads to a reduced first line,

$$\begin{aligned}&\underbrace{(Q_t+S_tK_t+C_t^\top Z_t)}_{=P_t}\partial_\theta x_t+\\ &\qquad\underbrace{\Phi_t+S_tK_t^\theta+C_t^\top Z_t^\theta}_{\overset{\text{def}}{=}\Lambda_t}+E_{t-1}^\top\partial_\theta\lambda_t=0\end{aligned} \tag{31}$$

where $\Lambda_t\in\mathbb{R}^{n_x\times n_\theta}$, thereby closing the recursion.

The recursion continues until reaching the initial stage, which gives the expression corresponding to $\nabla_{x_0}\mathcal{E}(x_0,\theta)$:

$$P_0x_0+\Lambda_0\theta+p_0.$$

Then, the sensitivity matrix $\partial_\theta \boldsymbol{\xi}$ can be extracted by a forward pass, iterating (30) for $t = 0, \ldots, N-1$.

Furthermore, the remaining value function parameters (σ_0, Σ_0) can be obtained by backward recursion:

Proposition 4. *The vector σ_0 and matrix Σ_0 from* (24) *can be computed as follows: let*

$$\sigma_N = \gamma_N \tag{32a}$$

$$\Sigma_N = \Gamma_N \tag{32b}$$

and for $t = N-1, \ldots, 0$,

$$\sigma_t = \sigma_{t+1} + \gamma_t + \Psi_t^\top k_t + \Lambda_{t+1}^\top a_t \tag{32c}$$

$$\Sigma_t = \Sigma_{t+1} + \Gamma_t + \Psi_t^\top K_t^\theta + \Lambda_{t+1}^\top M_t^\theta. \tag{32d}$$

Algorithm 2: Generalized Riccati recursion for parametric problems

```
Data: Cost and constraint matrices and vectors
      $Q_t, S_t, R_t, q_t, r_t, A_t, B_t, C_t, D_t, f_t, h_t$, parameters
      $\Phi_N, \gamma_N$
1  $P_N \leftarrow Q_N + \frac{1}{\mu} C_N^\top C_N$;
2  $p_N \leftarrow q_N + \frac{1}{\mu} C_N^\top \bar{h}_N$;
3  $\Sigma_N \leftarrow \Gamma_N$;
4  $\Lambda_N \leftarrow \Phi_N$;
5  $\sigma_N \leftarrow \gamma_N$;
   // Backward pass
6  for $t = N-1$ to $t = 0$ do
7      $[K_t^\theta, Z_t^\theta, \Omega_{t+1}^\theta, M_t^\theta] \leftarrow$ solve (29);
8      $P_t \leftarrow Q_t + S_t K_t + C_t^\top Z_t + A_t^\top \Omega_{t+1}$;
9      $p_t \leftarrow q_t + S_t k_t + C_t^\top \zeta_t + A_t^\top \omega_{t+1}$;
10     $\Sigma_t \leftarrow \Gamma_t + \Sigma_{t+1} + \Psi_t^\top K_t^\theta + \Lambda_{t+1}^\top M_t^\theta$;
11     $\Lambda_t \leftarrow \Phi_t + K_t^\top \Psi_t + M_t^\top \Lambda_{t+1}$;
12     $\sigma_t \leftarrow \sigma_{t+1} + \gamma_t + \Psi_t^\top k_t + \Lambda_{t+1}^\top a_t$;
13 $(x_0, \lambda_0, \theta) \leftarrow$ ComputeInitial();
   // Forward pass
14 for $t = 0$ to $N-1$ do
15     $u_t \leftarrow k_t + K_t x_t + K_t^\theta \theta$;
16     $v_t \leftarrow \zeta_t + Z_t x_t + Z_t^\theta \theta$;
17     $\lambda_{t+1} \leftarrow \omega_{t+1} + \Omega_{t+1} x_t + \Omega_{t+1}^\theta \theta$;
18     $x_{t+1} \leftarrow a_t + M_t x_t + M_t^\theta \theta$;
19 $v_N = \frac{1}{\mu}(\bar{h}_N + C_N x_N)$;
```

Algorithm: Bringing this all together, we outline a parametric generalized Riccati recursion in algorithm 2, which provides the solution of (23). Step 12 (ComputeInitial) of algorithm 2 is the procedure where the choice of parameter $\theta \in \mathbb{R}^{n_\theta}$, initial state x_0, and initial co-state λ_0 is made: θ and x_0 could be some fixed values, or all three could be decided jointly by satisfying an equation. This choice is application-dependent.

V. Parallelization of the LQ solver

In this section, we present a derivation for a parallelized variant of the algorithm introduced in section III, through the lens of solving parametric LQ problems.

We will split problem (1) into $J+1$ parts (or *legs*) where $1 \leqslant J < N$. Let $\mathcal{P} = \{0 = i_0 < i_1 < \cdots < i_J < N-1 < N\}$ be a set of partitioning indices for $[\![0, N]\!]$. We denote $\mathcal{I}_j = [\![i_j, i_{j+1} - 1]\!]$ for $j \leqslant J$ (where $i_{J+1} = N+1$), such that $[\![0, N]\!] = \bigcup_{j=0}^{J} \mathcal{I}_j$.

Each "leg" of the split problem, except for the terminal one, will be parameterized by the co-state λ_{i_j} (taking the place of θ in the previous section IV) which connects it to the next. In this respect, our algorithm differs from [34], where the parameter is the unknown value of the first state in the next leg.

A. 2-way split ($J = 1$)

For simplicity, we start by describing how an LQ problem can be split into two parts (the case $J = 1$) and explain how to merge them together.

The partition of $[\![0, N]\!]$ is $\mathcal{I}_0 \cup \mathcal{I}_1$ where $\mathcal{I}_0 = [\![0, i_1 - 1]\!]$ and $\mathcal{I}_1 = [\![i_1, N]\!]$, for some $1 \leqslant i_1 < N$. Denote $\tilde{\lambda}_1 = \lambda_{i_1}$ the co-state for the i_1-th dynamical constraint, which will be our splitting variable (taking the role of section IV's parameter θ), and $\tilde{x}_1 = x_{i_1}$ the corresponding state.

We proceed by splitting up the Lagrangian of the full-horizon problem:

$$\mathscr{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\nu}, \boldsymbol{\lambda}) = \mathscr{L}^0(\boldsymbol{x}_{\mathcal{I}_0}, \boldsymbol{u}_{\mathcal{I}_0}, \boldsymbol{\nu}_{\mathcal{I}_0}, \boldsymbol{\lambda}_{\mathcal{I}_0}; \tilde{\lambda}_1) + \tilde{\lambda}_1^\top E_{i_1 - 1} \tilde{x}_1 + \mathscr{L}^1(\boldsymbol{x}_{\mathcal{I}_1}, \boldsymbol{u}_{\mathcal{I}_1}, \boldsymbol{\nu}_{\mathcal{I}_1}, \boldsymbol{\lambda}_{\mathcal{I}_1}) \tag{33}$$

where $\mathscr{L}^0$ is the *parametric* Lagrangian for the first leg (running for $t \in \mathcal{I}_0$), *parameterized by* $\boldsymbol{\theta} = \tilde{\lambda}_1$:

$$\mathscr{L}^0(\boldsymbol{x}_{\mathcal{I}_0}, \boldsymbol{u}_{\mathcal{I}_0}, \boldsymbol{\nu}_{\mathcal{I}_0}, \boldsymbol{\lambda}_{\mathcal{I}_0}; \tilde{\lambda}_1) = \sum_{t=0}^{i_1 - 2} H_t(x_t, u_t, \nu_t, \lambda_{t+1}) + x_{t+1} E_t^\top \lambda_{t+1} + H_{i_1 - 1}(x_{i_1 - 1}, u_{i_1 - 1}, \nu_{i_1 - 1}, \tilde{\lambda}_1), \tag{34}$$

and the non-parametric Lagrangian $\mathscr{L}^1$ for the second leg (running for $t \in \mathcal{I}_1$):

$$\mathscr{L}^1(\boldsymbol{x}_{\mathcal{I}_1}, \boldsymbol{u}_{\mathcal{I}_1}, \boldsymbol{\nu}_{\mathcal{I}_1}, \boldsymbol{\lambda}_{\mathcal{I}_1}) = \sum_{t=i_1}^{N-1} H_t(x_t, u_t, \nu_t, \lambda_{t+1}) + x_{t+1} E_t^\top \lambda_{t+1} + \mathscr{L}_N(x_N, \nu_N). \tag{35}$$

We now define their respective value functions:

$$\mathcal{E}^0(x_0, \tilde{\lambda}_1) = \min_{\boldsymbol{x}, \boldsymbol{u}} \max_{\boldsymbol{\nu}, \boldsymbol{\lambda}} \mathscr{L}^0(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\nu}, \boldsymbol{\lambda}; \tilde{\lambda}_1) \tag{36a}$$

$$\mathcal{E}^1(\tilde{x}_1) = \min_{\boldsymbol{x}, \boldsymbol{u}} \max_{\boldsymbol{\nu}, \boldsymbol{\lambda}} \mathscr{L}^1(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\nu}, \boldsymbol{\lambda}) \tag{36b}$$

where the second value function does *not* depend on $\tilde{\lambda}_1$. Similarly to (23), we can define their dual-regularized value functions $\mathcal{E}_\mu^0$ and $\mathcal{E}_\mu^1$ (the former includes the $-\frac{\mu}{2}\|\tilde{\lambda}_1\|_2^2$ term). This allows us to write the full-horizon regularized LQ problem as the following saddle-point (assuming fixed x_0):

$$\min_{\tilde{x}_1} \max_{\tilde{\lambda}_1} \mathcal{E}_\mu^0(x_0, \tilde{\lambda}_1) + \tilde{\lambda}_1^\top E_{i_1 - 1} \tilde{x}_1 + \mathcal{E}_\mu^1(\tilde{x}_1). \tag{37}$$

We introduce the shorthands $\tilde{P}_1 = P_{i_1}$, $\tilde{p}_1 = p_{i_1}$, $\tilde{E}_1 = E_{i_1-1}$. The stationarity equations for this saddle-point are

$$\nabla_{\tilde{\lambda}_1} \mathcal{E}^0_\mu(x_0, \tilde{\lambda}_1) + \tilde{E}_1 \tilde{x}_1 = 0, \tag{38a}$$
$$\tilde{E}_1^\top \tilde{\lambda}_1 + \nabla_x \mathcal{E}^1_\mu(\tilde{x}_1) = 0, \tag{38b}$$

or, substituting the expression (24),

$$\begin{bmatrix} \Sigma_0 & \tilde{E}_1 \\ \tilde{E}_1^\top & \tilde{P}_1 \end{bmatrix} \begin{bmatrix} \tilde{\lambda}_1 \\ \tilde{x}_1 \end{bmatrix} = - \begin{bmatrix} \sigma_0 + \Lambda_0^\top x_0 \\ \tilde{p}_1 \end{bmatrix}. \tag{39}$$

The formulation can be further augmented if x_0 is a decision variable with initial constraint $G_0 x_0 + g_0 = 0$ associated with multiplier λ_0: the corresponding dual-regularized system is

$$\begin{bmatrix} -\mu I & G_0 & & \\ G_0^\top & P_0 & \Lambda_0 & \\ & \Lambda_0^\top & \Sigma_0 & \tilde{E}_1 \\ & & \tilde{E}_1^\top & \tilde{P}_1 \end{bmatrix} \begin{bmatrix} \lambda_0 \\ x_0 \\ \tilde{\lambda}_1 \\ \tilde{x}_1 \end{bmatrix} = - \begin{bmatrix} \bar{g}_0 \\ p_0 \\ \sigma_0 \\ \tilde{p}_1 \end{bmatrix}. \tag{40}$$

Forward pass: Once the above linear system is solved, we can reconstruct the full solution $(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\nu}, \boldsymbol{\lambda})$ by running a separate forward pass for each leg according to the equations in algorithm 2.

B. Generalization to $J \geqslant 2$

We now consider the case $J \geqslant 2$. For $0 \leqslant j \leqslant J$, we denote $\tilde{x}_j = x_{i_j}$ and $\tilde{\lambda}_j = \lambda_{i_j}$. Consider, for $j < J$, the value function $\mathcal{E}^j(\tilde{x}_j, \tilde{\lambda}_{j+1})$ associated with the subproblem running from time $t = i_j$ to $t = i_{j+1} - 1$:

$$\mathcal{E}^j(\tilde{x}_j, \tilde{\lambda}_{j+1}) = \min\max \sum_{t=i_j}^{i_{j+1}-2} H_t + \lambda_{t+1}^\top E_t x_{t+1} + H_{i_{j+1}-1}. \tag{41}$$

For $j = J$, the value function corresponding to the subproblem running for indices $t \in \mathcal{I}_J$ is

$$\mathcal{E}^J(\tilde{x}_J) = \min\max \sum_{t=i_J}^{N-1} H_t + \lambda_{t+1}^\top E_t x_{t+1} + \mathscr{L}_N. \tag{42}$$

We also define their proximal regularizations $(\mathcal{E}^j_\mu)_j$ as in (23).

Finally, the dual-regularized full-horizon LQ problem is equivalent to the saddle-point in $\tilde{\boldsymbol{x}} = (\tilde{x}_j)_{0 \leqslant j \leqslant J}$ and $\tilde{\boldsymbol{\lambda}}$:

$$\min_{\tilde{\boldsymbol{x}}} \max_{\tilde{\boldsymbol{\lambda}}} \sum_{j=0}^{J-1} \mathcal{E}^j_\mu(\tilde{x}_j, \tilde{\lambda}_{j+1}) + \tilde{\lambda}_{j+1}^\top \tilde{E}_{j+1} \tilde{x}_{j+1} + \mathcal{E}^J_\mu(\tilde{x}_J). \tag{43}$$

This problem also has a temporal structure, which leads to a block-tridiagonal system of equations.

We introduce the following notations, for $0 \leqslant j \leqslant J$ ($j < J$ for the last four):

$$\begin{aligned} \tilde{P}_j &= P_{i_j} & \tilde{p}_j &= p_{i_j} \\ \tilde{E}_{j+1} &= E_{i_{j+1}-1} & \tilde{\sigma}_j &= \sigma_{i_j} \\ \tilde{\Lambda}_j &= \Lambda_{i_j} & \tilde{\Sigma}_j &= \Sigma_{i_j} \end{aligned} \tag{44}$$

(in particular, $\tilde{p}_0 = p_0$, $\tilde{P}_0 = P_0$, and so on) so that, for $0 \leqslant j < J$,

$$\nabla_{\tilde{\lambda}_{j+1}} \mathcal{E}^j_\mu(\tilde{x}_j, \tilde{\lambda}_{j+1}) = \tilde{\sigma}_j + \tilde{\Lambda}_j^\top \tilde{x}_j + \tilde{\Sigma}_j \tilde{\lambda}_{j+1} \tag{45a}$$
$$\nabla_{\tilde{x}_j} \mathcal{E}^j_\mu(\tilde{x}_j, \tilde{\lambda}_{j+1}) = \tilde{p}_j + \tilde{\Lambda}_j \tilde{\lambda}_{j+1} + \tilde{P}_j \tilde{x}_j \tag{45b}$$

and

$$\nabla \mathcal{E}^J_\mu(\tilde{x}_J) = \tilde{p}_J + \tilde{P}_J \tilde{x}_J. \tag{45c}$$

Proposition 5. *The optimality conditions for eq.* (43) *are given by the following system of equations:*

$$\tilde{\sigma}_j + \tilde{\Lambda}_j^\top \tilde{x}_j + \tilde{\Sigma}_j \tilde{\lambda}_{j+1} + \tilde{E}_{j+1} \tilde{x}_{j+1} = 0, \ 0 \leqslant j < J \tag{46a}$$
$$\tilde{p}_j + \tilde{E}_j^\top \tilde{\lambda}_j + \tilde{P}_j \tilde{x}_j + \tilde{\Lambda}_j \tilde{\lambda}_{j+1} = 0, \ 1 \leqslant j < J \tag{46b}$$

and

$$\tilde{E}_J^\top \tilde{\lambda}_J + \tilde{P}_J \tilde{x}_J + \tilde{p}_J = 0. \tag{46c}$$

The system of equations can be summarized as the following sparse system:

$$\begin{array}{ccccccc|c} \lambda_0 & x_0 & \tilde{\lambda}_1 & \tilde{x}_1 & \tilde{\lambda}_2 & \cdots & \tilde{x}_J & \\ -\mu I & G_0 & & & & & & \bar{g}_0 \\ G_0^\top & P_0 & \Lambda_0 & & & & & p_0 \\ & \Lambda_0^\top & \Sigma_0 & \tilde{E}_1 & & & & \sigma_0 \\ & & \tilde{E}_1^\top & \tilde{P}_1 & \tilde{\Lambda}_1 & & & \tilde{p}_1 \\ & & & \tilde{\Lambda}_1^\top & \ddots & & & \tilde{\sigma}_1 \\ & & & & & \ddots & \tilde{E}_J & \vdots \\ & & & & & \tilde{E}_J^\top & \tilde{P}_J & \tilde{p}_J \end{array} \tag{47}$$

This system above can be solved by sparse $LDL^\top$ factorization, or by leveraging a specific block-tridiagonal algorithm. We will denote $\mathbb{M}$ the matrix in (47).

The overall method is summarized in algorithm 3.

Algorithm 3: Parallel condensation LQR

```
  Input:
1 for $j = 0, \dots, J$ in parallel do
      // Parameterize last stage of each leg
2     $F_{x,i_j} \leftarrow A_{i_j}^\top$;
3     $F_{u,i_j} \leftarrow B_{i_j}^\top$;
4     $\gamma_{i_j} \leftarrow f_{i_j}$;
      // Solve (43) parametrically in $\tilde{\lambda}_j$
5     Compute $(\tilde{P}_j, \tilde{\Lambda}_j, \tilde{\Sigma}_j, \tilde{p}_j, \tilde{\sigma}_j)$ using algorithm 2;
6 $(\tilde{x}_j)_j, (\tilde{\lambda}_j)_j \leftarrow$ solve consensus system (47);
7 for $0 = 1, \dots, J$ in parallel do
8     Compute the forward pass of (43) starting from $\tilde{x}_j$;
```

C. Block-tridiagonal algorithm for the reduced system

In this subsection, we present an efficient algorithm for solving the reduced linear system, exploiting its symmetric, block-tridiagonal structure by employing a block variant of the Thomas algorithm. The diagonal is $(-\mu I, \tilde{P}_0, \tilde{\Sigma}_0, \dots, \tilde{P}_J)$, and the superdiagonal is $(G_0, \tilde{\Lambda}_0, \tilde{E}_1, \dots, \tilde{E}_J)$. The elimination order will be from back-to-front, constructing a block-sparse $UDU^\top$ decomposition of the matrix $\mathbb{M}$. The resulting routine has linear complexity $\mathcal{O}(J)$ in the number of processors.

In a recent related paper, Jordana et al. [31] leverage the same algorithm to restate the classical Riccati recursion.

VI. Implementation in a nonlinear trajectory optimizer

We now consider a nonlinear discrete-time trajectory optimization problem with implicit system dynamics:

$$\min_{\boldsymbol{x},\boldsymbol{u}} J(\boldsymbol{x},\boldsymbol{u}) = \sum_{t=0}^{N-1} \ell_t(x_t,u_t) + \ell_N(x_N) \tag{48a}$$

$$\text{s.t. } x_0 = x^0 \tag{48b}$$

$$\phi_t(x_t,u_t,x_{t+1}) = 0 \tag{48c}$$

$$h_t(x_t,u_t) \leqslant 0 \tag{48d}$$

$$h_N(x_T) \leqslant 0. \tag{48e}$$

The implicit discrete dynamics $\phi_t(x_t,u_t,x_{t+1}) = 0$ is often a discretization scheme for an ODE $\dot{x} = f(t,x,u)$ or implicit ODE $f(t,x,u,\dot{x}) = 0$ (e.g. implicit Runge-Kutta methods).

Following [28], we use a proximal augmented Lagrangian scheme to solve this problem. This solver implements an outer (proximal, augmented Lagrangian) loop which iteratively solves a family of proximal LQ problems (6) *instead* of the initial LQ problem (1). The coefficients of the problem are obtained from the derivatives of (48) with the following equivalences:

$$\begin{aligned}
&A_t = \phi_{x,t} \; B_t = \phi_{u,t} \; E_t = \phi_{y,t} \\
&C_t = h_{x,t} \; D_t = h_{u,t} \\
&f_t = \phi_t(x_t,u_t,x_{t+1}) \\
&q_t = \ell_{x,t} + A_t^\top \lambda_{t+1} + C_t^\top \nu_t + E_{t-1}^\top \lambda_t \\
&r_t = \ell_{u,t} + B_t^\top \lambda_{t+1} + D_t^\top \nu_t \\
&Q_t = \ell_{xx,t} + \lambda_{t+1} \cdot \phi_{xx,t} + \nu_t \cdot h_{xx,t} + \lambda_t \cdot \phi_{yy,t-1} \\
&R_t = \ell_{uu,t} + \lambda_{t+1} \cdot \phi_{uu,t} + \nu_t \cdot h_{uu,t} \\
&S_t = \ell_{xu,t} + \lambda_{t+1} \cdot \phi_{xu,t} + \nu_t \cdot h_{xu,t},
\end{aligned}$$

All these quantities are directly obtained from applying a semi-smooth primal-dual Newton step, as detailed in [30][3]

We then implemented our parallel algorithm for solving the proximal LQ. Our implementation will be open-sourced upon acceptance of the paper. It is then straightforward to adapt an implementation of [29] by replacing the proximal LQ solver by the parallel formulation. This also makes it possible to fairly compare the new algorithm with the original formulation.

VII. Discussion

In the introduction, we mentioned a few prior methods for parallelizing the resolution of system (5), and made a distinction between indirect and direct methods – the terms "indirect" and "direct" are used in the linear algebra sense. In this section, we will provide a more detailed discussion of these prior methods to distinguish with our own.

In [24] and [35], multiple-shooting formulations for nonlinear DTOC problems are shown to enable parallel nonlinear rollouts in the forward pass. There, multiple-shooting defects are used at specific knots of the problem (called *shooting states*) to split the horizon into segments where, once the shooting state is known (through a serial linear rollout), nonlinear rollouts on each segment can be performed in parallel. Our method, however, can parallelize both the backward *and* (linear) forward passes, but no strategy for a nonlinear rollout has been discussed in this paper. In light of the hybrid rollout strategy of [24], parallel nonlinear rollouts between each "leg" could be used in a nonlinear solver after applying our method to compute a linear policy and linear rollout in parallel.

Method	**Type**	Notes
Multiple-shooting [24, 35]	Direct	Only the forward pass is parallel.
PCG [1, 8]	Indirect	Adapted to GPUs. Exploits structure through sparse preconditioner.
Gauss-Seidel [18, 42]	Indirect	Not meant to solve (1), but produce iterations for (48). [18] explicitly considers equality constraints.
ADMM [48]	Indirect	ADMM splitting costs, dynamical, and state-control constraints.
State linkage [34]	Direct	$\boldsymbol{\lambda}$ obtained by least-squares.
State-control linkage [38]	Direct	Results in subproblems similar to [52].
Co-state split [52, 39]	Direct	[39] works on the value function parameters. [52] considers constraints (through nullspace/QR decomposition). Both condense into another instance of (1).
Co-state split (**Ours**)	Direct	Condenses into state/co-state system (43) instead of another instance of (1). Explicitly considers equality constraints, implicit dynamics, dual proximal regularization.

TABLE I
Comparison table of multiple methods for parallel computation of the LQ step.

Indirect methods. As for indirect methods for solving (5), we distinguish a few subclasses of methods. These methods all work towards parallelizing the entire computation of a linear search direction for (48) by approximating a solution for (1). One is given by preconditioned conjugate gradient (PCG) methods such as [1, 8] which are well-suited to GPUs, exploiting the block-banded structure to find appropriate preconditioners. In particular [1, 8] reduce (5), through a tailored Schur complement, to a system in the co-states $\boldsymbol{\lambda}$ which is solved iteratively. A major assumption there is that both cost Hessian matrices (Q_t, R_t) are positive definite (there are no (S_t) cross-terms, and equality constraints are not considered). Another idea suited to nonlinear OC is to re-use a previous set of value function parameters, as proposed by [42] using the multiple-shooting forward sweep from [24]. A similar idea is to adapt Gauss-Seidel iteration, as in [18], where the Riccati recursion is modified to use a previous iteration's linear relation between λ_{t+1} and x_t. Finally, [48] propose an instance of ADMM [7] which alternatively iterates between optimizing cost and projecting onto the dynamical constraints.

Direct methods. In comparison, our method is squarely in

[3]We have neglected two second-order terms corresponding to the derivatives $\partial^2_{u_t x_{t+1}} \mathscr{L} = \lambda_{t+1} \cdot \phi_{uy,t}(x_t,u_t,x_{t+1})$ and $\partial^2_{x_t x_{t+1}} \mathscr{L} = \lambda_{t+1} \cdot \phi_{xy,t}(x_t,u_t,x_{t+1})$. They could be added to the block-sparse factorization of section III-C with further derivations which are outside the scope of this paper.

the realm of direct methods for solving (5). These methods exploit problem structure when applying classical factorization or (block) Gaussian elimination procedures. As presented in section V, our method splits the problem at specific co-states at stages $t \in \mathcal{P} = (i_j)_j$, and condenses it into a saddle-point over the splitting states and co-states $(x_t, \lambda_t)_{t\in\mathcal{P}}$. In contrast, Wright [52] condenses the problem into another MPC-like subproblem which also includes controls (and is thus over $(x_t, u_t, \lambda_t)_{t\in\mathcal{P}}$). Nielsen and Axehill [39] achieve a similar set of subproblems, by partial condensing (eliminating states and keeping their unknown controls (u_t)) followed by reduction through SVD and parametrizing with respect to the value function parameters – this is similar to parametrizing with respect to co-states. Laine and Tomlin [34] condense the problem into an (overdetermined) linear system in the co-states through a linkage constraint in the states. [52, 38, 39] show that their constructions – which yield MPC subproblems in the same form as (1), can be iterated to further reduce each subproblem. In our setting, the linear subproblem (47) does not have that same structure (such that our construction from section V cannot be iterated), however, it is still possible to leverage or design a parallel structure-exploiting routine (generic sparse e.g., [44], or specific for block-tridiagonal systems).

The different methods that have been discussed are summarized in Table I.

VIII. Experiments

We have implemented the algorithms presented in this paper in C++ using the Eigen [25] linear algebra library and the OpenMP API [12] for parallel programming. This implementation has been added to our optimal control framework ALIGATOR[4]. It is the authors' aim to improve its efficiency in the future.

A. Cyclic LQ problem

Figures 1 and 2 show trajectories for sample cyclic LQ problems, which are formulated in appendix D.

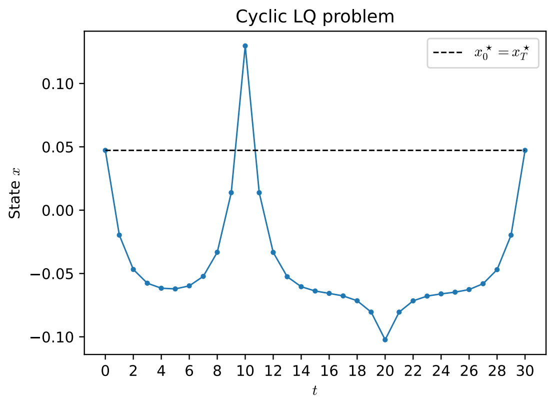

Fig. 1. LQ problem with cyclical constraint $x_0 = x_{30}$,in one dimension. No other initial condition was provided for x_0.

[4]https://github.com/Simple-Robotics/aligator/

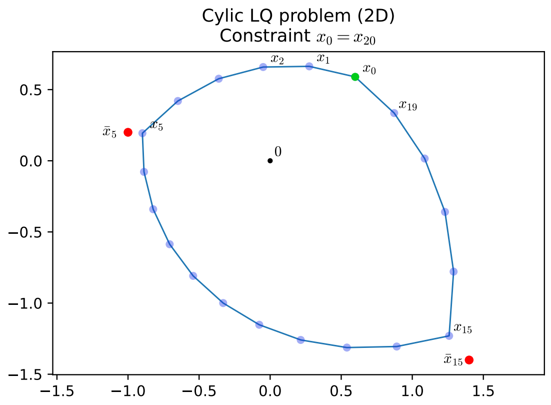

Fig. 2. Cyclic LQ problem in the 2D plane. Cost function $\ell(x,u) = 10^{-3}\|x\|^2 + \|u\|^2$ except at for $t \in \{5, 15\}$ where $\ell(x,u) = 0.2\|x - \bar{x}_t\|^2 + \|u\|^2$.

B. Synthetic benchmark

To assess the speedups our implementation of the parallel algorithm could achieve, we implemented a synthetic benchmark of problems with different horizons ranging from $T = 16, \ldots, 2048$. The benchmark was run on an Apple Mac Studio M1 Ultra, which has 20 cores (16-P cores, 4 E-cores). The resulting timings are given in Figure 3. These results show that our implementation is not able to reach 100% efficiency (defined as the ratio of the speedup to the number of cores). Instead, we obtain between 50-60% efficiency for the longer horizons, but this decreases for shorter problems. This might be due to the overhead of solving (47), and that of dispatching data between cores. Greater efficiency would certainly be obtained with a more refined implementation (optimizing memory allocation, cache-friendliness).

C. Nonlinear trajectory optimization

Computation of robot dynamical quantities (joint acceleration, frame jacobians…) is provided by the Pinocchio [10, 11] rigid-body dynamics library. The TALOS benchmark and NMPC experiments were run on a Dell XPS laptop with an Intel i9-13900k CPU (8 P-cores and 16 E-cores).

1) TALOS locomotion benchmarks: We consider a whole-body trajectory optimization problem on a TALOS [47] humanoid robot with constrained 6D contacts. For this robot, the state and control dimensions are $n_x = 57$ and $n_u = 22$ respectively. The robot follows a user-defined contact sequence with feet references going smoothly from one contact to the next. State and controls are regularized towards an initial static half-sitting position and $u = 0$ respectively. Single-foot support time T_{ss} is set to 4 times double-support time T_{ds}. We consider three instances of the problem with different time horizons, encompassing two full steps of the robot. The problem is discretized using the semi-implicit Euler scheme with timestep $\Delta t = 10\,\text{ms}$. In fig. 4, our proximal solver with various parallelization settings is compared against the feasibility-prone DDP from the CROCODDYL library [36].

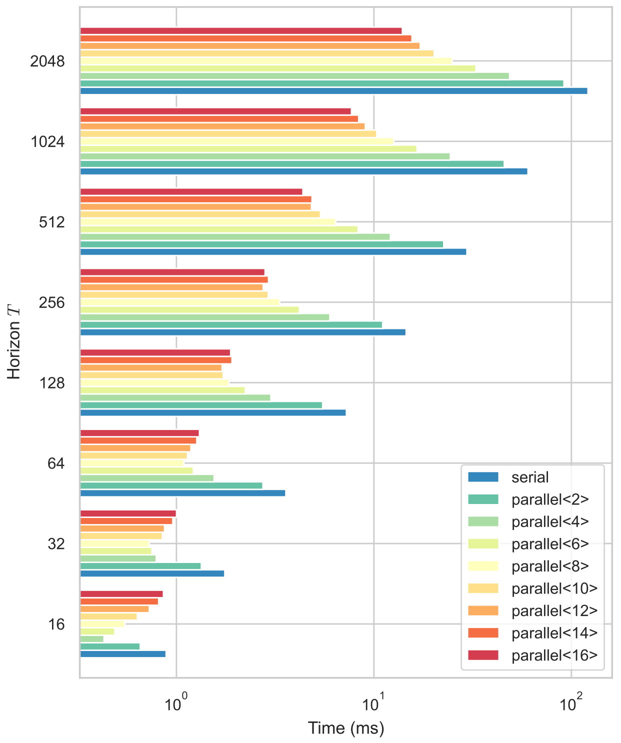

Fig. 3. Timings for a backward-forward sweep of the solver on a synthetic benchmark.

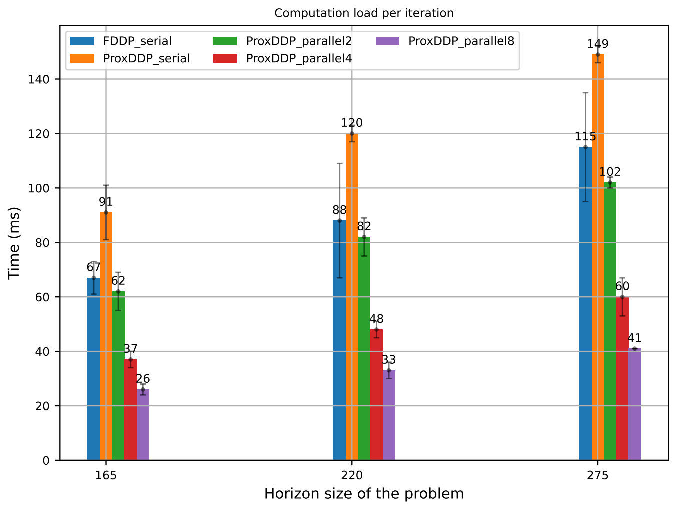

Fig. 4. C++ benchmarks of a trajectory optimization problem involving two forward steps with the whole-body model of the TALOS robot, with single support time T_{ss} set to 0.6 s, 0.8 s and 1 s from left to right. Each instance is run 40 times on every solver to produce a mean and standard deviation. Here, the Turboboost feature on the CPU was disabled and clock speed fixed to 2200 MHz.

2) Constrained NMPC on TALOS*:* In this subsection, we leverage our proximal solver to perform whole-body nonlinear MPC on the humanoid robot TALOS in simulation, similarly to what is achieved in [15] on real hardware. The problem remains the same as in the previous subsection, except that

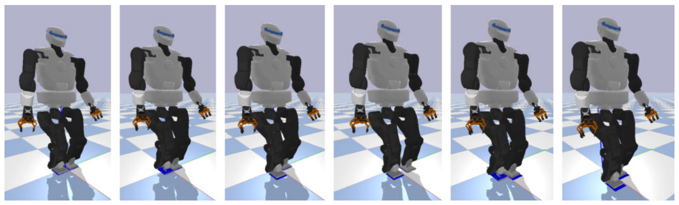

Fig. 5. Snapshot of a PyBullet [14] simulation featuring TALOS walking with pre-defined feet trajectories (blue rectangles). The Bullet simulation timestep is set to 1 ms.

we add equality constraints at the end of each flying phase to ensure that foot altitude and 6D velocity are zero at impact time (7 constraints per foot). Horizon window is set to 0.5 s with a timestep of 10 ms for a total of $N = 50$ steps. A snapshot of this experiment is displayed in fig. 5.

In order to test the benefits of parallelization in the NMPC setting, the walking motion experiment has been run with varying numbers of threads. The timing results of this experiment are shown in fig. 6. When using 4 threads, computation time is cut by a factor 2 with respect to the serial algorithm. However, above 4 threads, the speedup stagnates.

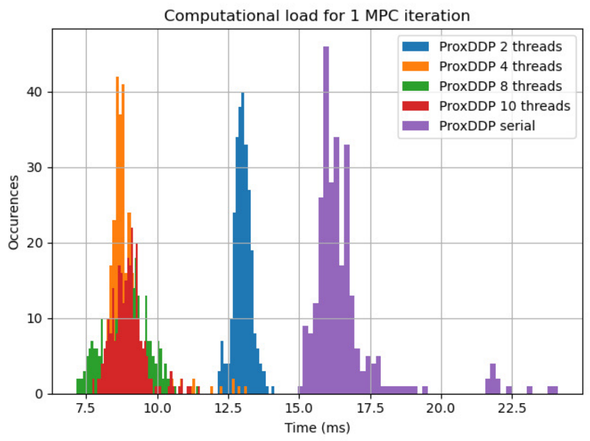

Fig. 6. Comparison of the performances of parallel and serial proximal algorithms on the TALOS walking MPC. The histogram shows the distribution of time per iteration for a two-step motion. The simulation was written using PyBullet, and the proximal solver was called through its Python API.

3) Parallel NMPC on SOLO-12*:* To demonstrate the capabilities of our solver in a real-world setting, we integrated it inside a whole-body nonlinear MPC framework to control an actual torque-controlled quadruped robot, SOLO-12, adapting the framework of Assirelli et al. [3].

The framework formulates an optimal control problem (OCP) with a state of dimension $n_x = 37$ $(12 + 12$ joint positions/velocities and $7 + 6$ base pose/velocity) and controls of size $n_u = 12$. Akin to the experiment on TALOS, the robot follows a user-defined contact sequence, yet with no predefined reference foot trajectories. The horizon is set to 0.96 s, with a 12 ms timestep, resulting in a discrete-time horizon of $N = 80$.

The above framework is divided into two parts, running on separate computers communicating via local Ethernet:

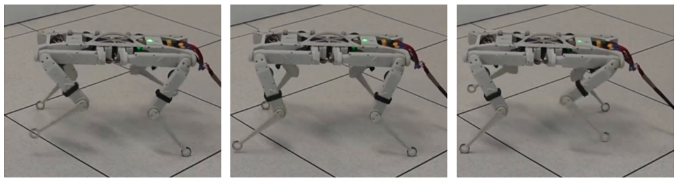

Fig. 7. SOLO-12 walking on flat ground.

- a high-level OCP solver runs on a powerful desktop,
- a lower-level, high-frequency controller on a laptop with the necessary drivers for real-time robot control. It interpolates the Riccati gains and feedforward controls between two MPC cycles, producing a reference torque and joint trajectory for the robot at 1 kHz.

The time budget for solving the OCP is approximately 9 ms. The MPC has a frequency of 12 ms (same as the discretization step of the OCP), communication requires 2 ms (round-trip), and 1 ms is left for margin and additional computations. This limited budget, coupled with the problem's long horizon and dimensionality, justifies the need for parallelization.

The solve times for a single iteration of the OCP in the MPC loop were measured over 20,000 MPC cycles in the experiment. Table II summarizes some statistics.

No. of threads	2	4	8	10	12
Mean time (ms)	10.3	6.0	4.6	4.6	4.4
Std. dev (ms)	1.0	1.2	0.97	0.90	0.85

TABLE II
MEAN TIME (AND STANDARD DEVIATION) FOR THE MPC EXPERIMENT ON SOLO-12. STATISTICS COMPUTED OVER 20,000 CONTROL CYCLES.

Computer specifications: Apple Mac Studio M1 Ultra desktop with 20 cores (16 P-cores and 4 E-cores), and Dell XPS laptop with an Intel Core i7-10510U CPU.

IX. CONCLUSION

In this paper, we have discussed the proximal-regularized LQ problem, as a subproblem in nonlinear MPC solvers, and have introduced a serial Riccati-like algorithm with a block-sparse resolution method which allows efficient resolution of this proximal iteration subproblem. Furthermore, we extended this method to solving parametric LQ problems and then leveraged this to formulate a parallel version of the initial algorithm.

We demonstrated that the parallel algorithm is able to handle complex tasks on high-dimensional robots with long horizons, even for real-time NMPC scenarios. Still, while we are able to reach good overall execution times, the benchmarks suggest our implementation is not able to reach high parallel efficiency as of yet. This suggests an avenue for further work on this implementation and further experimental validation and benchmarking. Furthermore, the timings on the tested systems open the door to experimenting with more complex motions, assessing the benefits of either longer time horizons or higher-frequency MPC schemes.

ACKNOWLEDGEMENTS

WJ thanks Ajay Sathya for his insights into block-matrix factorization in proximal settings. This work was supported in part by the French government under the management of Agence Nationale de la Recherche (ANR) as part of the "Investissements d'avenir" program, references ANR-19-P3IA-0001 (PRAIRIE 3IA Institute), ANR-19-p3IA-0004 (ANITI 3IA Institute) and ANR-22-CE33-0007 (INEXACT), the European project AGIMUS (Grant 101070165), the Louis Vuitton ENS Chair on Artificial Intelligence and the Casino ENS Chair on Algorithmic and Machine Learning.

REFERENCES

[1] Emre Adabag, Miloni Atal, William Gerard, and Brian Plancher. Mpcgpu: Real-time nonlinear model predictive control through preconditioned conjugate gradient on the gpu, September 2023.

[2] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. page 12.

[3] Alessandro Assirelli, Fanny Risbourg, Gianni Lunardi, Thomas Flayols, and Nicolas Mansard. Whole-body mpc without foot references for the locomotion of an impedance-controlled robot.

[4] Dimitri Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[5] Oumayma Bounou, Jean Ponce, and Justin Carpentier. Leveraging proximal optimization for differentiating optimal control solvers. In *IEEE 62nd Conference on Decision and Control (CDC)*, Singapore, Singapore, December 2023.

[6] Stephen Boyd. Proximal algorithms. page 113.

[7] Stephen Boyd. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2010. ISSN 1935-8237, 1935-8245. doi: 10.1561/2200000016.

[8] Xueyi Bu and Brian Plancher. Symmetric stair preconditioning of linear systems for parallel trajectory optimization, September 2023.

[9] James Bunch and Linda Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Mathematics of Computation - Math. Comput.*, 31:163–163, January 1977. doi: 10.1090/S0025-5718-1977-0428694-0.

[10] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *IEEE International Symposium on System Integrations (SII)*, 2019.

[11] Justin Carpentier, Rohan Budhiraja, and Nicolas Mansard. Proximal and sparse resolution of constrained dynamic equations. In *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, July 2021.

ISBN 978-0-9923747-7-8. doi: 10.15607/RSS.2021.XVII.017.

[12] Rohit Chandra, Leo Dagum, David Kohr, Ramesh Menon, Dror Maydan, and Jeff McDonald. *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[13] Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software*, 35(3):1–14, October 2008. ISSN 0098-3500, 1557-7295. doi: 10.1145/1391989.1391995.

[14] Erwin Coumans and Yunfei Bai. *PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning*. 2016/2020.

[15] Ewen Dantec, Maximilien Naveau, Pierre Fernbach, Nahuel Villa, Guilhem Saurel, Olivier Stasse, Michel Taix, and Nicolas 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)*, pages 638–644, November 2022. doi: 10.1109/Humanoids53995.2022.10000129.

[16] Ewen Dantec, Michel Taïx, and Nicolas Mansard. First order approximation of model predictive control solutions for high frequency feedback. *IEEE Robotics and Automation Letters*, 7(2):4448–4455, April 2022. ISSN 2377-3766. doi: 10.1109/LRA.2022.3149573.

[17] Timothy A. Davis. Algorithm 849: A concise sparse cholesky factorization package. *ACM Transactions on Mathematical Software*, 31(4):587–591, December 2005. ISSN 0098-3500. doi: 10.1145/1114268.1114277.

[18] Haoyang Deng and Toshiyuki Ohtsuka. A highly parallelizable newton-type method for nonlinear model predictive control. *IFAC-PapersOnLine*, 51:349–355, January 2018. doi: 10.1016/j.ifacol.2018.11.058.

[19] M. Diehl, H.G. Bock, H. Diedam, and P.-B. Wieber. Fast direct multiple shooting algorithms for optimal robot control. In Moritz Diehl and Katja Mombaur, editors, *Fast Motions in Biomechanics and Robotics*, volume 340, pages 65–93. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-36118-3. doi: 10.1007/978-3-540-36119-0_4.

[20] 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*, 63(1):23–38, October 1989. ISSN 1573-2878. doi: 10.1007/BF00940728.

[21] Farbod Farshidian et al. OCS2: An open source library for optimal control of switched systems. [Online]. Available: https://github.com/leggedrobotics/ocs2.

[22] Gianluca Frison. Algorithms and methods for high-performance model predictive control. page 345.

[23] Markus Giftthaler and Jonas Buchli. A projection approach to equality constrained iterative linear quadratic optimal control. *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 61–66, November 2017. doi: 10.1109/HUMANOIDS.2017.8239538.

[24] Markus Giftthaler, Michael Neunert, M. Stäuble, J. Buchli, and M. Diehl. A family of iterative gauss-newton shooting methods for nonlinear optimal control. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018. doi: 10.1109/IROS.2018.8593840.

[25] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[26] Taylor A. Howell, Brian E. Jackson, and Zachary Manchester. Altro: A fast solver for constrained trajectory optimization. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7674–7679, Macau, China, November 2019. IEEE. ISBN 978-1-72814-004-9. doi: 10.1109/IROS40897.2019.8967788.

[27] David H. Jacobson and David Q. Mayne. *Differential Dynamic Programming*. American Elsevier Publishing Company, 1970. ISBN 978-0-444-00070-5.

[28] Wilson Jallet, Antoine Bambade, Nicolas Mansard, and Justin Carpentier. Constrained differential dynamic programming: A primal-dual augmented lagrangian approach. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Kyoto, Japan, October 2022. doi: 10.1109/IROS47612.2022.9981586.

[29] Wilson Jallet, Nicolas Mansard, and Justin Carpentier. Implicit differential dynamic programming. In *2022 International Conference on Robotics and Automation (ICRA)*, Philadelphia, United States, May 2022. IEEE Robotics and Automation Society. doi: 10.1109/ICRA46639.2022.9811647.

[30] Wilson Jallet, Antoine Bambade, Etienne Arlaud, Sarah El-Kazdadi, Nicolas Mansard, and Justin Carpentier. Proxddp: Proximal constrained trajectory optimization, 2023.

[31] Armand Jordana, Sébastien Kleff, Avadesh Meduri, Justin Carpentier, Nicolas Mansard, and Ludovic Righetti. Stagewise implementations of sequential quadratic programming for model-predictive control, December 2023.

[32] Sarah Kazdadi, Justin Carpentier, and Jean Ponce. Equality constrained differential dynamic programming. In *ICRA 2021 - IEEE International Conference on Robotics and Automation*, May 2021.

[33] Forrest Laine and Claire Tomlin. Efficient computation of feedback control for equality-constrained lqr. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6748–6754, May 2019. doi: 10.1109/ICRA.2019.8793566.

[34] Forrest Laine and Claire Tomlin. Parallelizing lqr computation through endpoint-explicit riccati recursion. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1395–1402, December 2019. doi: 10.1109/CDC40024.2019.9029974.

[35] He Li, Wenhao Yu, Tingnan Zhang, and Patrick M. Wensing. A unified perspective on multiple shooting in differential dynamic programming. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Sys-*

tems (IROS), pages 9978–9985, October 2023. doi: 10.1109/IROS55552.2023.10342217.

[36] Carlos Mastalli, Rohan Budhiraja, Wolfgang Merkt, Guilhem Saurel, Bilal Hammoud, Maximilien Naveau, Justin Carpentier, Ludovic Righetti, Sethu Vijayakumar, and Nicolas Mansard. Crocoddyl: An efficient and versatile framework for multi-contact optimal control. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2536–2542, May 2020. doi: 10.1109/ICRA40945.2020.9196673.

[37] David Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1): 85–95, January 1966. ISSN 0020-7179. doi: 10.1080/00207176608921369.

[38] Isak Nielsen and Daniel Axehill. An ô (log n) parallel algorithm for newton step computation in model predictive control. *IFAC Proceedings Volumes*, 47(3): 10505–10511, January 2014. ISSN 1474-6670. doi: 10.3182/20140824-6-ZA-1003.01577.

[39] Isak Nielsen and Daniel Axehill. A parallel structure exploiting factorization algorithm with applications to model predictive control. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 3932–3938, December 2015. doi: 10.1109/CDC.2015.7402830.

[40] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, New York, 2nd ed edition, 2006. ISBN 978-0-387-30303-1.

[41] Alex Oshin, Matthew D Houghton, Michael J. Acheson, Irene M. Gregory, and Evangelos Theodorou. Parameterized differential dynamic programming. In *Robotics: Science and Systems XVIII*. Robotics: Science and Systems Foundation, June 2022. ISBN 978-0-9923747-8-5. doi: 10.15607/RSS.2022.XVIII.046.

[42] Brian Plancher and Scott Kuindersma. A performance analysis of parallel differential dynamic programming on a gpu. In Marco Morales, Lydia Tapia, Gildardo Sánchez-Ante, and Seth Hutchinson, editors, *Algorithmic Foundations of Robotics XIII*, volume 14, pages 656–672. Springer International Publishing, Cham, 2020. ISBN 978-3-030-44050-3 978-3-030-44051-0. doi: 10.1007/978-3-030-44051-0_38.

[43] R. T. Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1 (2):97–116, 1976. ISSN 0364-765X.

[44] Olaf Schenk, Klaus Gärtner, Wolfgang Fichtner, and Andreas Stricker. Pardiso: A high-performance serial and parallel sparse linear solver in semiconductor device simulation. *Future Generation Computer Systems*, 18 (1):69–78, September 2001. ISSN 0167-739X. doi: 10.1016/S0167-739X(00)00076-5.

[45] Akshay Srinivasan and Emanuel Todorov. Graphical newton. 2017.

[46] Akshay Srinivasan and Emanuel Todorov. Computing the newton-step faster than hessian accumulation. *arXiv:2108.01219 [cs, eess, math]*, August 2021.

[47] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, J.-P. Laumond, L. Marchionni, H. Tome, and F. Ferro. Talos: A new humanoid research platform targeted for industrial applications. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 689–695, November 2017. doi: 10.1109/HUMANOIDS.2017.8246947.

[48] Georgios Stathopoulos, Tamas Keviczky, and Yang Wang. A hierarchical time-splitting approach for solving finite-time optimal control problems. *2013 European Control Conference (ECC)*, pages 3089–3094, July 2013. doi: 10.23919/ECC.2013.6669702.

[49] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems.*, pages 4906–4913, October 2012. ISBN 978-1-4673-1737-5. doi: 10.1109/IROS.2012.6386025.

[50] Lander Vanroye, Joris De Schutter, and Wilm Decré. A generalization of the riccati recursion for equality-constrained linear quadratic optimal control, June 2023.

[51] Stephen J Wright. Solution of discrete-time optimal control problems on parallel computers. *Parallel Computing*, 16(2-3):221–237, December 1990. ISSN 01678191. doi: 10.1016/0167-8191(90)90060-M.

[52] Stephen J. Wright. Partitioned dynamic programming for optimal control. *SIAM Journal on Optimization*, 1 (4):620–642, November 1991. ISSN 1052-6234. doi: 10.1137/0801037.

Appendix

A. Refresher on Riccati recursion

In this appendix, we provide a self-contained and concise refresher on Riccati recursion. We will consider an unconstrained, unregularized linear-quadratic problem

$$\min_{\boldsymbol{x},\boldsymbol{u}} J(\boldsymbol{x},\boldsymbol{u}) \overset{\text{def}}{=} \sum_{t=0}^{N-1} \ell_t(x_t,u_t) + \ell_N(x_N) \tag{50a}$$

$$\text{s.t. } x_{t+1} = A_t x_t + B_t u_t + f_t \tag{50b}$$

$$x_0 = x^0 \tag{50c}$$

where the cost functions ℓ_t and ℓ_N are the same as (2), and $x^0 \in \mathbb{R}^{n_x}$ is the problem's initial data. Compared to problem (1), this problem has *no contraints*, an *explicit initial condition* and *explicit system dynamics*.

Optimality conditions. The KKT conditions for (50) read as follow: there exist co-states $\{\lambda_t\}_{0\leqslant t\leqslant N}$ such that

$$\lambda_N = Q_N x_N + q_N, \tag{51a}$$

$$0 = x^0 - x_0 \tag{51b}$$

as well as, for $0 \leqslant t < N$,

$$Q_t x_t + S_t u_t + q_t + A_t^\top \lambda_{t+1} = \lambda_t, \tag{51c}$$

$$S_t^\top x_t + R_t u_t + r_t + B_t^\top \lambda_{t+1} = 0 \tag{51d}$$

$$A_t x_t + B_t u_t + f_t - x_{t+1} = 0. \tag{51e}$$

Recursion. The recursion hypothesis is as follows: there exists a semidefinite positive *cost-to-go* matrix P_t and vector p_t such that

$$\lambda_t = P_t x_t + p_t. \tag{52}$$

This is true for $t = N$ owing to the terminal optimality condition (51a).

Now, let $0 \leqslant t < N$ such that the recursion hypothesis is true for $t+1$, i.e. there exist (P_{t+1}, p_{t+1}) such that $\lambda_{t+1} = P_{t+1} x_{t+1} + p_{t+1}$. Then, plugging (51e) into this leads to

$$\lambda_{t+1} = P_{t+1}(A_t x_t + B_t u_t + f_t) + p_{t+1}$$

which with eqs. (51c) and (51d) leads into

$$\begin{bmatrix} \widehat{Q}_t & \widehat{S}_t \\ \widehat{S}_t^\top & \widehat{R}_t \end{bmatrix} \begin{bmatrix} x_t \\ u_t \end{bmatrix} + \begin{bmatrix} \widehat{q}_t \\ \widehat{r}_t \end{bmatrix} = \begin{bmatrix} \lambda_t \\ 0 \end{bmatrix}, \tag{53}$$

where we have denoted

$$\widehat{Q}_t = Q_t + A_t^\top P_{t+1} A_t, \tag{54a}$$

$$\widehat{S}_t = S_t + A_t^\top P_{t+1} B_t, \tag{54b}$$

$$\widehat{R}_t = R_t + B_t^\top P_{t+1} B_t, \tag{54c}$$

$$\widehat{q}_t = q_t + A_t^\top (p_{t+1} + P_{t+1} f_t), \tag{54d}$$

$$\widehat{r}_t = r_t + B_t^\top (p_{t+1} + P_{t+1} f_t). \tag{54e}$$

The following step in the Riccati recursion is to apply the Schur complement lemma to (53) (see appendix B), requiring that $\widehat{R}_t \succ 0$. This leads to

$$\lambda_t = (\widehat{Q}_t - \widehat{S}_t \widehat{R}_t^{-1} \widehat{S}_t^\top) x_t + \widehat{q}_t - \widehat{S}_t^\top \widehat{R}_t^{-1} \widehat{r}_t \tag{55a}$$

as well as the familiar linear feedback equation

$$u_t = k_t + K_t x_t \tag{55b}$$

where we denote by $K_t = -\widehat{R}_t^{-1} \widehat{S}_t^\top$ and $k_t = -\widehat{R}_t^{-1} \widehat{r}_t$ the classic *feedback* and *feedforward* gains. The recursion is then closed by setting

$$P_t \overset{\text{def}}{=} \widehat{Q}_t - \widehat{S}_t \widehat{R}_t^{-1} \widehat{S}_t^\top \in \mathbb{S}_{n_x}(\mathbb{R}) \tag{56a}$$

and

$$p_t \overset{\text{def}}{=} \widehat{q}_t - \widehat{S}_t \widehat{R}_t^{-1} \widehat{r}_t. \tag{56b}$$

To go further, we direct the reader to an application of this recursion in the context of nonlinear control for robotics in [49], which introduced the popular *iterative LQR* (iLQR) algorithm.

B. Schur complement lemma

In this appendix, we will provide an overview of the Schur complement lemma.

1) 2x2 block linear system: We consider a symmetric 2×2 block linear system

$$\begin{bmatrix} G & J^\top \\ J & -\Lambda \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = - \begin{bmatrix} b \\ c \end{bmatrix} \tag{57}$$

where G (resp. Λ) is a $n \times n$ (resp. $m \times m$) symmetric matrix, $b \in \mathbb{R}^n$, $c \in \mathbb{R}^m$, and $J \in \mathbb{R}^{m \times n}$.

Symmetric indefinite systems such as (57) can be solved by straightforward *LU*, *LDL*$^\top$ or Bunch-Kaufman [9] decomposition. Specific methods exist for sparse matrices [17, 13].

Assuming nonsingular Λ, the Schur complement in x, solves z as a function of x:

$$z = \Lambda^{-1}(c + Jx) \tag{58}$$

and leads to an equation in x:

$$(G + J^\top \Lambda^{-1} J)x = -(b + J^\top \Lambda^{-1} c). \tag{59}$$

Here, $G_\Lambda = G + J^\top \Lambda^{-1} J$ is called the *Schur matrix*. If, furthermore, we have that $G \succeq 0$ and $\Lambda \succ 0$, then the Schur matrix is positive semidefinite.

In some applications (e.g. constrained forward dynamics [11]), G exhibits sparsity and the the complement in z is chosen instead, computing G^{-1} – this corresponds to the partial minimization $\min_x f(x,z)$.

2) Variant: lemma for proposition 3: Consider the following parametric saddle-point problem:

$$\psi^\star(z) = \min_x \max_y f(x,y,z) \tag{60}$$

where f is the following quadratic:

$$f(x,y,z) = \frac{1}{2} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \begin{bmatrix} Q & S & C^\top \\ S^\top & R & D^\top \\ C & D & H \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + q^\top x + r^\top y + c^\top z \tag{61}$$

with appropriate assumptions on (Q,S,R) to ensure that f is convex-concave (namely, $Q \succeq 0$ and that $R \preceq 0$) and

$$M = \begin{bmatrix} Q & S \\ S^\top & R \end{bmatrix} \tag{62}$$

is nonsingular. The first-order conditions satisfied by a solution $(x^\star, y^\star)$ of (60) read:

$$\begin{bmatrix} Q & S & C^\top \\ S^\top & R & D^\top \end{bmatrix} \begin{bmatrix} x^\star \\ y^\star \\ z \end{bmatrix} + \begin{bmatrix} q \\ r \end{bmatrix} = 0. \tag{63}$$

By nonsingularity of M, it holds

(i) that $(x^\star, y^\star)$ is linear in z with

$$\begin{bmatrix} x^\star \\ y^\star \end{bmatrix} = -M^{-1} \left(\begin{bmatrix} q \\ r \end{bmatrix} + \begin{bmatrix} C^\top \\ D^\top \end{bmatrix} z \right), \tag{64}$$

and, as a consequence,

(ii) that $\psi^\star$ is a quadratic function of z: there exist a matrix W and vector w such that

$$\psi^\star(z) = \frac{1}{2} z^\top W z + w^\top z, \tag{65}$$

and (W, w) is given by

$$
\begin{aligned}
W &= H - \begin{bmatrix} C & D \end{bmatrix} M^{-1} \begin{bmatrix} C^\top \\ D^\top \end{bmatrix}, \\
w &= c - \begin{bmatrix} C & D \end{bmatrix} M^{-1} \begin{bmatrix} q \\ r \end{bmatrix}.
\end{aligned} \tag{66}
$$

C. Thomas algorithm for block-tridiagonal matrices

Consider a block-sparse linear system

$$
\mathbb{M} \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} C_1 \\ \vdots \\ C_N \end{bmatrix} \tag{67}
$$

where the X_i and C_i are $n_i \times m$ matrices, and

$$
\mathbb{M} = \begin{bmatrix} A_0 & B_1 & & & \\ B_1^\top & A_1 & B_2 & & \\ & B_2^\top & A_2 & \ddots & \\ & & \ddots & & B_N \\ & & & B_N^\top & A_N \end{bmatrix} \tag{68}
$$

is a block-triadiagonal matrix, where $A_i \in \mathbb{R}^{n_i \times n_i}$, $B_i \in \mathbb{R}^{n_{i-1} \times n_i}$.

A quick derivation of the backward-forward algorithm stems by considering an appropriate block $UDU^\top$ factorization, where we prescribe

$$
U = \begin{bmatrix} I & \mathsf{U}_1 & & \\ & I & & \\ & & \ddots & \\ & & I & \mathsf{U}_N \end{bmatrix}, \ D = \begin{bmatrix} \mathsf{D}_0 & & & \\ & \mathsf{D}_1 & & \\ & & \ddots & \\ & & & \mathsf{D}_N \end{bmatrix}.
$$

Writing $\mathbb{M} = UDU^\top$, it appears that a sufficient and necessary condition is that (U, D) satisfy

$$
A_N = \mathsf{D}_N \tag{69a}
$$
$$
B_i = \mathsf{U}_i \mathsf{D}_i, \ 1 \leqslant i \leqslant N \tag{69b}
$$
$$
A_i = \mathsf{D}_i + \mathsf{U}_{i+1} \mathsf{D}_{i+1} \mathsf{U}_{i+1}^\top, \ 0 \leqslant i < N, \tag{69c}
$$

which determines (U, D) uniquely. The second and third equations can be combined together as $\mathsf{D}_i = A_i - B_{i+1} \mathsf{D}_{i+1}^{-1} B_{i+1}^\top$.

Furthermore, we use (U, D) to solve the initial system as $UDU^\top X = C$, which can be split up as $U^\top X = Z$ and $UDZ = C$. The second equation can be solved by working backwards (since U is block upper-triangular):

$$
Z_N = \mathsf{D}_N^{-1} C_N \tag{70a}
$$
$$
Z_i = \mathsf{D}_i^{-1} (C_i - B_{i+1} Z_{i+1}), \ 0 \leqslant i < N \tag{70b}
$$

and then solve the first equation for X, working forwards:

$$
X_0 = Z_0 \tag{71a}
$$
$$
X_{i+1} = Z_{i+1} - \mathsf{U}_{i+1}^\top X_i, \ 0 \leqslant i < N. \tag{71b}
$$

D. Formulating LQ problems with cyclic constraints

Assume the LQ problem to solve is similar to (1) with a cyclical constraint

$$
x_N - x_0 = 0
$$

and, for simplicity, no other path constraints (meaning $n_c = 0$). We introduce for this constraint an additional multiplier $\theta \in \mathbb{R}^{n_x}$ as a parameter. The full problem Lagrangian is

$$
\mathscr{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}; \theta) = \mathscr{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\lambda}) + \theta^\top (x_N - x_0).
$$

The terminal stage Lagrangian becomes $\mathscr{L}_N(x_N, \theta) = \ell_N(x_N) + \theta^\top x_N$. In this setting, our previous derivations apply with with $\Phi_N = I$. After condensing into the parameterized value function $\mathcal{E}(x_0, \theta)$, we can solve for (x_0, θ) by solving the min-max problem

$$
\max_\theta \min_{x_0} \mathcal{E}(x_0, \theta) - \theta^\top x_0 \tag{72}
$$

leading to the system

$$
\begin{bmatrix} P_0 & \Lambda_0 - I \\ (\Lambda_0 - I)^\top & \end{bmatrix} \begin{bmatrix} x_0 \\ \theta \end{bmatrix} = - \begin{bmatrix} p_0 \\ \sigma_0 \end{bmatrix}. \tag{73}
$$

Other topologies such as directed acyclic graphs (DAG) have been explored in the literature [45, 46]. Such a discussion would be out of the scope of this paper, but it is our view that several algorithms in this vein can be restated using parametric Lagrangians and by formulating partial min-max problems.

E. Details for the block-sparse factorization in Section III-C

We present here the details of how we propose to solve (12)-(14) by exploiting its block-sparse structure, replacing the less efficient recursion initially proposed in [28]. The first step is isolating the equations, in (12), satisfied by the co-state and next state (λ_2, x_2). They are:

$$
\begin{array}{cc|c} \lambda_2 & x_2 & \\ \hline -\mu I & E_1 & \bar{f}_1 + A_1 x_1 + B_1 u_1 \\ E_1^\top & P_2 & p_2 \end{array}. \tag{74}
$$

This system could be solved by Schur complement in either variables λ_2 or x_2. One way requires computing a decomposition of $P_2 + \frac{1}{\mu} E_1^\top E_1$, which is numerically unstable if μ is small. The other way requires P_2 to be nonsingular, and computing the inverse of $\mu I + E_1 P_2^{-1} E_1^\top$ – although this would be more stable, assuming nonsingular P_2 can be a hurdle in practical applications (typically, if the terminal cost only applies to part of the state space such as joint velocities).

1) Substitution by E_2: We approach this another way, by assuming that the dynamics matrix E_1 is nonsingular. This means the substitution $\check{x}_2 = -E_1 x_2$ is well-defined, and that we can define $\check{P}_2 = E_1^{-T} P_2 E_1^{-1}$ and $\check{p}_2 = -E_1^{-T} p_2$.

Then, the system (74) is equivalent to

$$
\begin{array}{cc|c} \lambda_2 & \check{x}_2 & \\ \hline -\mu I & -I & \bar{f}_1 + A_1 x_1 + B_1 u_1 \\ -I & \check{P}_2 & \check{p}_2 \end{array} \tag{75}
$$

which by substituting $\check{x}_2 = \bar{f}_1 + A_1 x_1 + B_1 u_1 - \mu\lambda_2$ reduces to the symmetric linear system

$$(\mu\check{P}_2 + I)\lambda_2 = \check{p}_2 + \check{P}_2(\bar{f}_1 + A_1 x_1 + B_1 u_1). \tag{76}$$

Denote $\Upsilon = I + \mu\check{P}_2$, and set $\mathcal{V}_2 = \Upsilon^{-1}\check{P}_2$ and $v_2 = \Upsilon^{-1}(\check{p}_2 + \check{P}_2\bar{f}_1)$. Then, it holds that

$$\lambda_2 = v_2 + \mathcal{V}_2(A_1 x_1 + B_1 u_1).$$

Substitution into (12) yields equations for (x_1, u_1, v_1):

$$\begin{array}{ccc|c} x_1 & u_1 & v_1 & \\ \widehat{Q}_1 & \widehat{S}_1 & C_1^\top & \widehat{q}_1 + E_0^\top\lambda_1 \\ \widehat{S}_1^\top & \widehat{R}_1 & D_1^\top & \widehat{r}_1 \\ C_1 & D_1 & -\mu I & \bar{h}_1 \end{array} \tag{77}$$

where $\widehat{Q}_1, \widehat{S}_1, \widehat{R}_1, \widehat{q}_1, \widehat{r}_1$ are given by the familiar equations:

$$\begin{aligned} \widehat{Q}_1 &= Q_1 + A_1^\top \mathcal{V}_2 A_1, & \widehat{q}_1 &= q_1 + A_1^\top v_2, \\ \widehat{R}_1 &= R_1 + B_1^\top \mathcal{V}_2 B_1, & \widehat{r}_1 &= r_1 + B_1^\top v_2, \\ \widehat{S}_1 &= S_1 + A_1^\top \mathcal{V}_2 B_1. \end{aligned} \tag{78}$$

2) Multiplier and next-state update: The closed-loop multiplier update is

$$\lambda_2 = v_2 + \mathcal{V}_2 B_1 k_1 + \mathcal{V}_2(A_1 + B_1 K_1)x_1 = \omega_2 + \Omega_2 x_1 \tag{79}$$

and the next state update is

$$\begin{aligned} x_2 &= -E_1^{-1}\check{x}_2, \\ \check{x}_2 &= (\bar{f}_1 + B_1 k_1 - \mu\omega_2) + (A_1 + B_1 K_1 - \mu\Omega_2)x_1 \\ &= a_1 + M_1 x_1. \end{aligned} \tag{80}$$

3) Final substitution: Now, our goal is to eliminate (u_1, v_1) from the system, expressing them as a function of x_1. Denote $\widehat{\mathcal{K}}_1$ the lower-right 2×2 block

$$\widehat{\mathcal{K}}_1 = \begin{bmatrix} \widehat{R}_1 & D_1^\top \\ D_1 & -\mu I \end{bmatrix}, \tag{81}$$

which is a nonsingular symmetric matrix due to the dual regularization block $-\mu I$. Then, by Schur complement, we obtain (u_1, v_1) as a feedback

$$\begin{bmatrix} u_1 \\ v_1 \end{bmatrix} = -\widehat{\mathcal{K}}_1^{-1}\left(\begin{bmatrix} \widehat{r}_1 \\ \bar{h}_1 \end{bmatrix} + \begin{bmatrix} \widehat{S}_1^\top \\ C_1 \end{bmatrix} x_1\right) \tag{82}$$

which can be rewritten as

$$u_1 = k_1 + K_1 x_1, \; v_1 = \zeta_1 + Z_1 x_1. \tag{83}$$

Then, it holds that the cost-to-go matrix and gradient introduced in (16) also satisfy

$$P_1 = \widehat{Q}_1 + \widehat{S}_1 K_1 + C_1^\top Z_1 \tag{84a}$$
$$p_1 = \widehat{q}_1 + \widehat{S}_1 k_1 + C_1^\top \zeta_1. \tag{84b}$$

The full block-sparse algorithm is summarized in Algorithm 4.

Algorithm 4: Block-sparse factorization for the stage KKT equations

```
Data: Cost and constraint matrices
      $Q_1, S_1, R_1, q_1, r_1, A_1, B_1, E_1, \bar{f}_1, C_1, D_1, \bar{h}_1$,
      cost-to-go matrix and vector $P_2, p_2$
1  Compute $E_1^{-1}$;                         // using, e.g., LU
2  $\check{P}_2 \leftarrow E_1^{-T} P_2 E_1^{-1}$;
3  $\check{p}_2 \leftarrow -E_1^{-T} p_2$;
4  $\Upsilon \leftarrow \mu\check{P}_2 + I$ ;
5  $\mathcal{V}_2 \leftarrow \Upsilon^{-1}\check{P}_2$;
6  $v_2 \leftarrow \Upsilon^{-1}(\check{p}_2 + \check{P}_2\bar{f}_1)$;
7  Set $(\widehat{Q}_1, \widehat{S}_1, \widehat{R}_1, \widehat{q}_1, \widehat{r}_1)$ according to (78);
8  Set $\widehat{\mathcal{K}}_1 \leftarrow \begin{bmatrix} \widehat{R}_1 & D_1^\top \\ D_1 & -\mu I \end{bmatrix}$;
9  Compute $\widehat{\mathcal{K}}_1^{-1}$ (using, e.g., LDL$^\mathrm{T}$);
   // Solve (82)
10 $\begin{bmatrix} k_1 \\ \zeta_1 \end{bmatrix} \leftarrow -\widehat{\mathcal{K}}_1^{-1}\begin{bmatrix} \widehat{r}_1 \\ \bar{h}_1 \end{bmatrix}$;
11 $\begin{bmatrix} K_1 \\ Z_1 \end{bmatrix} \leftarrow -\widehat{\mathcal{K}}_1^{-1}\begin{bmatrix} \widehat{S}_1^\top \\ C_1 \end{bmatrix}$;
12 $P_1 \leftarrow \widehat{Q}_1 + \widehat{S}_1 K_1 + C_1^\top Z_1$;
13 $p_1 \leftarrow \widehat{q}_1 + \widehat{S}_1 k_1 + C_1^\top \zeta_1$;
```

F. Bound for the parallel algorithm's speedup

In this appendix, we etch a derivation for the complexity of the serial and parallel backward passes of our algorithms 1, 2 and 3, which we will compare to find an upper-bound for the expected speedup.

Consider the stage system (14) in the backward pass of algorithm 1. We denote by:

- $C_{\mathrm{fac}} = \mathcal{O}((n_x + n_u + n_c)^3)$ the complexity of the factorization step
- $C_{\mathrm{col}} = \mathcal{O}((n_x + n_u + n_c)^2)$ the complexity of the right-hand side solve for a single column.

Then, the primal-dual gains are recovered with complexity $(n_x + 1)C_{\mathrm{col}}$. Furthermore, for the parametric algorithm, solving the parameter gains $(K_t^\theta, Z_t^\theta, \Omega_{t+1}^\theta, M_t^\theta)$ in (29) for a parameter of dimension n_θ has complexity $C_{\mathrm{param}} = n_\theta C_{\mathrm{col}}$.

The complexity of the backward pass in the serial algorithm 1 is thus

$$T^{\mathrm{b}}_{\mathrm{serial}} = N(C_{\mathrm{fac}} + (n_x + 1)C_{\mathrm{col}}). \tag{85}$$

Now, consider the parallel algorithm executed over $J+1$ ($J \geqslant 1$) legs. Each stage in the backward pass for legs $0 \leqslant j < J$ has complexity $C_{\mathrm{fac}} + (2n_x + 1)C_{\mathrm{col}}$ (the last leg still has standard stage complexity $C_{\mathrm{fac}} + (n_x + 1)C_{\mathrm{col}}$). Meanwhile, the consensus system (43) has complexity $C_{\mathrm{cons}} \sim \frac{2J+1}{3} n_x^3$ (assuming Cholesky decompositions with complexity $n^3/3$). We assume an equal split across the J legs, such that $i_{j+1} - i_j = N/(J+1)$. The parallel time complexity for this backward pass is thus

$$T^{\mathrm{b}}_{\mathrm{parallel}} = \underbrace{\tfrac{N}{J+1}(C_{\mathrm{fac}} + (2n_x + 1)C_{\mathrm{col}})}_{\text{parallel section}} + C_{\mathrm{cons}}. \tag{86}$$

We plot the speedup ratio $T^{\mathrm{b}}_{\mathrm{serial}}/T^{\mathrm{b}}_{\mathrm{parallel}}$ of serial to parallel execution times for a chosen dimension (n_x, n_u, n_c) in fig. 8. The graph shows early departure of this ratio from the "perfect" speedup (which would be $T^{\mathrm{b}}_{\mathrm{parallel}} = JT^{\mathrm{b}}_{\mathrm{serial}}$).

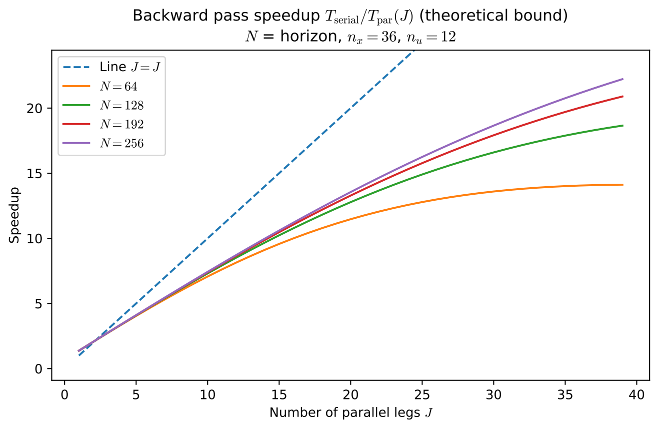

Fig. 8. Theoretical speedup derived from the bounds, with varying problem horizon N and number of parallel legs/processors J. The problem dimensions correspond to the Solo-12 system in section VIII.