An Adaptive Restart Heavy-Ball Projected Primal-Dual Method for Solving Constrained Linear Quadratic Optimal Control Problems

An Adaptive Restart Heavy-Ball Projected Primal-Dual Method for Solving Constrained Linear Quadratic Optimal Control Problems

Y.J.J. Heuts, G.P. Padilla, M.C.F. Donkers

Abstract— **This paper presents a fast and flexible projected primal-dual method for solving linear quadratic optimal control problems with box constraints. Using a specific preconditioning, the algorithm achieves dead-beat convergence for unconstrained problems and has fast convergence for constrained problems. Accelerated convergence is obtained by applying a heavy-ball method to accelerate the projected primal-dual algorithm. In order to avoid missing critical points due to high momentum, an adaptive restarting procedure is used to slow the algorithm down if the solution diverges. Furthermore, convergence is proven by representing the algorithm as a Lur'e-type dynamic system and applying LaSalle's invariance principle to show the fixed point is asymptotically stable. The resulting algorithm is simple, while also achieving competitive computational times.**

I. Introduction

Traditionally, (online) optimal control has been reserved for controlling linear systems with slow sample times over small control horizons. Namely, as the sampling rate and/or the control horizon increases, the requirements on computation times for the solver for the underlying optimization problem increase. Applications, where optimal control is used in an embedded microcontroller for systems with high sampling rates, can be found, e.g., [1], where typically control horizons are kept short, or applications, where time horizons increase, can be found in, e.g., [2], where computation times are typically large. Another trend is to apply numerical optimal control to nonlinear systems or problems with integer decisions. In the former case, sequential quadratic programming, e.g., [3] can be used to solve the problem, while in the latter branch-and-bound methods, e.g., [4] can be applied. In both cases, a linear-quadratic sub-problem has to be solved repeatedly. This calls for solvers for linear quadratic (LQ) optimal control problems (OCPs) that have a low computational effort that scales well when the number of states/inputs and time horizon increases.

Typically, discrete-time LQ OCPs can be reformulated as a (sparse or dense) quadratic programming problem [5] for which off-the-shelf solvers exist, e.g., CPLEX [6] or Matlab's quadprog. Although these solvers work well, these solvers do not exploit the underlying structure present in the OCP. Furthermore, they can be difficult to implement on microcontrollers. An alternative to this can be found in open-source dedicated solvers, such as OSQP [7] and FBStab [8]. These methods are based on operator splitting methods [9], [10] and show state-of-the-art performance.

While operator splitting methods show good performance on large-scale problems, because they often only need gradient information, they do not have the same convergence rates as interior-point and active-set methods used in CPLEX and quadprog. The limited convergence rate of operator splitting methods can be improved either using the heavy ball-method by Polyak [11] or applying Nesterov's acceleration [12]. Alternatively, hybrid methods can be used that combine operator splitting with a second-order method. This is done in OSQP and FBStab, wherein the case of OSQP, operator splitting methods are used to converge close to the optimal solution and a polishing strategy is used to refine the solution.

In this paper, we propose a fast and flexible solver for LQ OCPs based on operator splitting methods. In particular, we propose a heavy-ball and preconditioned projected primal-dual method. Our approach is based on a specific preconditioning of the algorithm, which causes the solver to show resemblance to a 'Gauss-Seidel method', in which the solution of the primal update is directly fed into the update of the dual variable. Furthermore, the preconditioning incorporates second-order information. This preconditioning is similar to the one proposed in [13] and causes the algorithm to obtain a solution in 1 iteration in absence of state and input constraints. Because of the specific structure of the OCP, matrices are sparse and we leverage on solvers for sparse linear systems. The resulting algorithm can be implemented in Matlab using only 30 lines of code.

We focus for now on a LQ-OCP with diagonal weights and so-called box constraints, which causes the used projection to become a simple saturation. Because of the proposed preconditioning and projection, we can analyze convergence by presenting the algorithm as a Lur'e-type dynamic system, e.g., a linear time-invariant system with a static nonlinearity induced by a saturation. We show using LaSalle's invariance principle that convergence of the algorithm is guaranteed. To improve convergence of the algorithm, a heavy-ball method is applied, where we increase and restart the momentum whenever needed, using an approach adapted from [14]. Finally, the paper presents two simulation examples that show that little tuning is needed and that the tuning is robust, i.e, holds for multiple cases or for varying horizon lengths. Finally, we show that the solver can compete with state-of-the-art solvers in terms of computation time.

This work has received financial support from the Horizon 2020 programme of the European Union under the grants 'Efficient and environmental friendly LONG distance poweRtrain for heavy dUty trucks aNd coaches' (LONGRUN-874972).

The authors are with the Dept. of Electrical Eng. of Eindhoven University of Technology, Netherlands. E-mail: {y.j.j.heuts, g.p.padilla,m.c.f.donkers}@tue.nl

II. Problem formulation

In this section, we will introduce the finite horizon discrete-time LQ OCP with state and input constraints that is considered in this paper. Later, we will reformulate this OCP into a highly structured static optimization problem. The structure of this formulation will be exploited in Section III to introduce a simple and computationally efficient method that finds the solution to the original discrete-time OCP.

A. Constrained Linear Quadratic Optimal Control Problem

Consider a discrete-time horizon $\mathcal{K} = \{0, \dots, K\}$, where $K \in \mathbb{N}$ is the horizon length. For a given controllable discrete-time dynamical system with states $x_k \in \mathbb{R}^{n_x}$ and inputs $u_k \in \mathbb{R}^{n_u}$, $k \in \mathcal{K}$, the discrete-time LQ OCP under consideration is given by

$$\begin{aligned}
\min_{x_k, u_k} \ & \tfrac{1}{2}(x_K - r_K)^\top P (x_K - r_K) \\
& + \tfrac{1}{2} \sum_{k=0}^{K-1} (x_k - r_k)^\top Q (x_k - r_k) + u_k^T R u_k && \text{(1a)} \\
\text{s.t.} \quad & x_{k+1} = A x_k + B u_k && \text{(1b)} \\
& \underline{x} \leqslant x_k \leqslant \bar{x} && \text{(1c)} \\
& \underline{u} \leqslant u_k \leqslant \bar{u}, && \text{(1d)}
\end{aligned}$$

for a given initial condition $x_0 = x_{\text{init}}$ and given reference trajectory r_k for all $k \in \mathcal{K}$, and where A and B describe the dynamics of a controllable system. The weighing matrices P, Q and R are diagonal positive-definite matrices, $\overline{x}$ and $\underline{x}$ are the state upper and lower bounds, respectively, and $\overline{u}$ and $\underline{u}$ are the input upper and lower bounds, respectively. We assume that the given problem is feasible within the given constraints.

B. A Sparse QP Formulation

The use of static optimization techniques to obtain the solution to the discrete-time OCP is common practice, see, e.g., [5], [15]. The solution to the OCP in (1) can be found by solving a quadratic program (QP), where the static formulation can have great impact on the performance of the optimization method. Generally, QP formulations can be divided in condensed and sparse formulations. Condensed approaches [5] have a reduced number of decision variables, as a result of eliminating the state in the formulation. Consequently, matrices are dense, which results in the computational complexity and memory requirements scaling cubically and quadratically with respect to the horizon length [15]. This motives the choice of a QP sparse formulation in this paper.

The OCP in (1) can be rewritten into the sparse formulation, whereby considerable amounts of zero entries are ensured in the matrices, as

$$\begin{aligned}
\min_{w} \quad & \tfrac{1}{2}\omega^\top \mathbf{G} \omega + \mathbf{F}^\top \omega + \imath_\Omega(\omega) && \text{(2a)} \\
\text{s.t.} \quad & \mathbf{A}\omega - \mathbf{b} = 0, && \text{(2b)}
\end{aligned}$$

where $\omega = \begin{bmatrix} x^T & u^T \end{bmatrix}^T$, in which $x = \begin{bmatrix} x_0^\top \dots x_K^\top \end{bmatrix}^\top$, and $u = \begin{bmatrix} u_0^\top \dots u_{K-1}^\top \end{bmatrix}^\top$, $\imath_\Omega(\omega)$ denotes the indicator function satisfying $\imath_\Omega(\omega) = 0$ if $\omega \in \Omega$ and $\imath_\Omega(\omega) = \infty$ if $\omega \notin \Omega$, where Ω is the box-constrained feasible set

$$\Omega := \{\omega \in \mathbb{R}^{(K+1)n_x + K n_u} \mid \underline{\omega} \leqslant \omega \leqslant \overline{\omega}\}, \tag{3}$$

in which $\underline{\omega}$ and $\overline{\omega}$ are defined similar to ω. The matrices in the cost function (1a) are fitted into sparse matrices

$$\mathbf{G} = \operatorname{diag}(Q, \dots, Q, P, R, \dots, R), \tag{4}$$

$$\mathbf{F} = [-r_0^\top Q, \dots, -r_{K-1}^\top Q, -r_K^\top P, 0, \dots, 0]^\top, \tag{5}$$

and the discrete-time dynamics are captured in the equality constraint (2b), where $\mathbf{A} = \begin{bmatrix} \Gamma_x & \Gamma_u \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} x_{\text{init}}^\top & 0 & \dots & 0 \end{bmatrix}^\top$ and

$$\Gamma_x = \begin{bmatrix} I & 0 & \dots & 0 \\ A & -I & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ & 0 & A & -I \end{bmatrix}, \quad \Gamma_u = \begin{bmatrix} 0 & \dots & 0 \\ B & 0 & \vdots \\ 0 & \ddots & 0 \\ 0 & \dots & B \end{bmatrix}. \tag{6}$$

It should be noted that formulation (2) is highly structured. For instance, $\mathbf{G}$ is a diagonal matrix and $\mathbf{A}$ is a sparse block-banded-matrix. This structure and sparsity in (2) will also be exploited in the following section to propose a computationally efficient optimization algorithm.

III. Heavy-Ball Projected Primal-Dual Algorithm

In this section, we will introduce the projected primal-dual method after which we show that by choosing a specific preconditioning, the projection can be expressed by a max-min operator. Lastly, we show that the algorithm converges for this specific choice of preconditioning.

The optimal solution of the static optimization problem is obtained using the following non-smooth Lagrangian function

$$L(\omega, \lambda) = \tfrac{1}{2}\omega^\top \mathbf{G}\omega + \mathbf{F}^\top \omega + \lambda^\top (\mathbf{A}\omega - \mathbf{b}) + \imath_\Omega(\omega). \tag{7}$$

The optimal solution satisfies the KKT conditions, given by

$$0 \in \begin{bmatrix} \mathbf{G} & \mathbf{A}^\top \\ -\mathbf{A} & 0 \end{bmatrix} \begin{bmatrix} \omega \\ \lambda \end{bmatrix} + \begin{bmatrix} \mathbf{F} + \mathcal{N}_\Omega(\omega) \\ \mathbf{b} \end{bmatrix}, \tag{8}$$

where $\mathcal{N}_\Omega$ is the normal cone operator satisfying $\mathcal{N}_S(\omega) = \{v \in \mathbb{R}^n \mid \sup_{z \in \Omega} v^\top (z - \omega) \leqslant 0\}$ if $\omega \in \Omega$ and $\mathcal{N}_\Omega(\omega) = \varnothing$ if $\omega \notin \Omega$. We propose in this paper a projected primal-dual method with heavy-ball, in which a KKT point is obtained by finding a fixed point of

$$\begin{cases} \omega^{i+1} = \operatorname{proj}_\Omega^{\Phi_\omega}\{\omega^i - \Phi_\omega(\mathbf{G}\omega^i + \mathbf{F} + \mathbf{A}^\top \lambda^i)\} \\ \qquad\qquad + \beta_\omega(\omega^i - \omega^{i-1}) \\ \lambda^{i+1} = \lambda^i + \Phi_\lambda(\mathbf{A}\omega^{i+1} - \mathbf{b}) + \beta_\lambda(\lambda^i - \lambda^{i-1}), \end{cases} \tag{9}$$

for index $i \in \mathbb{N}$, some positive definite preconditioning matrix Φ_ω, nonsingular matrix Φ_λ and some nonnegative scalars $\beta_\omega, \beta_\lambda$. In (9), $\operatorname{proj}_\Omega^{\Phi_\omega}(v) = \arg\min_{x \in \Omega} (x - v)^\top \Phi_\omega^{-1} (x - v)$ denotes a (weighted) Euclidean projection. The algorithm is based on a preconditioned projected Primal-Dual method with Heavy-Ball acceleration and Gauss-Seidel update of the dual variable λ.

In the lemma below, we will establish that (9) has a fixed point $(\omega^\star, \lambda^\star)$ that solves (8). Then, in the next section, we will make a specific choice for Φ_ω, Φ_λ, β_ω and β_λ and prove that (ω^i, λ^i) converge to the fixed point $(\omega^\star, \lambda^\star)$.

Lemma 1. *Assume* (2) *is feasible and assume Φ_ω is a positive-definite matrix and Φ_λ is nonsingular. Then, the iteration* (9) *has a fixed point $(\omega^{i+1}, \lambda^{i+1}) = (\omega^i, \lambda^i) = (\omega^\star, \lambda^\star)$ for $i \to \infty$ that satisfies the KKT conditions* (8)*. This fixed point $\omega^\star = [(x^\star)^\top, (u^\star)^\top]^\top$ is the optimal solution to* (1)*.*

Proof. First, observe that $\mathrm{proj}_\Omega^\Phi(v)$, which is a proximal operator [10], is the inverse relation of $v + \Phi\mathcal{N}_\Omega(v)$, meaning that (9) is equivalent to

$$\begin{cases} \omega^{i+1} \in \omega^i - \Phi_\omega(\mathbf{G}\omega^i + \mathbf{F} + \mathbf{A}^\top\lambda^i + \mathcal{N}_\Omega(\omega^{i+1})) \\ \qquad\qquad +\beta_\omega(\omega^i - \omega^{i-1}) \\ \lambda^{i+1} = \lambda^i + \Phi_\lambda(\mathbf{A}\omega^{i+1} - \mathbf{b}) + \beta_\lambda(\lambda^i - \lambda^{i-1}). \end{cases} \tag{10}$$

Now the fixed point of (10), i.e., where $(\omega^{i+1}, \lambda^{i+1}) = (\omega^i, \lambda^i) = (\omega^\star, \lambda^\star)$, satisfies the KKT conditions (8).

To complete the proof, recall that we assumed the problem is feasible, meaning that a fixed point exists that satisfies the KKT conditions. Furthermore, since problem (2) is convex and only has linear qualities, this fixed point is a (unique) minimum to (2). Since (2) is an equivalent reformulation to (1), the point $\omega^\star = [x^\star, u^\star]^\top$ is the optimal solution to (1), which completes the proof. □

A. Step-Size/Preconditioning Matrix Selection

After presenting the projected primal-dual method, and showing that the fixed point of this method satisfies the KKT conditions, a specific choice for the step-size and preconditioning is given to achieve convergence. The step size used in our developments below is given by $\Phi_\omega = \mathbf{G}^{-1}$, $\Phi_\lambda = \alpha(\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top)^{-1}$, $\beta_\omega = 0$ and $\beta_\lambda = \beta$, with $\alpha \in (0, 1]$ and $\beta \in [0, 1]$. We will show below that the iteration converges in this case.

The consequence of this choice of the preconditioning matrix $\Phi_\omega = \mathbf{G}^{-1}$, where $\mathbf{G}$ is assumed to be diagonal in this paper, is that $\mathrm{proj}_\Omega^{\mathbf{G}^{-1}}(v) = \max\{\underline{\omega}, \min\{\overline{\omega}, v\}\}$. Consequently, for these choices of preconditioning matrices, (9) reduces to

$$\begin{cases} \omega^{i+1} = \max\{\underline{\omega}, \min\{\overline{\omega}, \{-\mathbf{G}^{-1}\mathbf{F} - \mathbf{G}^{-1}\mathbf{A}^\top\lambda^i\}\} \\ \lambda^{i+1} = \lambda^i + \alpha(\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top)^{-1}(\mathbf{A}\omega^{i+1} - \mathbf{b}) \\ \qquad\quad +\beta(\lambda^i - \lambda^{i-1}). \end{cases} \tag{11}$$

It will be shown in Section IV that this method can be implemented numerically in an extremely efficient manner, even for large-dimensional problems.

It should be noted that this form of preconditioning has been proposed before in [13], but our main motivation is the fact that in absence of constraints (i.e., $\Omega = \mathbb{R}^n$) and for $\alpha = 1$ and $\beta = 0$, the algorithm converges in 1 step. Namely, in absence of the max-min operation in (11), the expression for ω^{i+1} can be substituted in the second expression, leading to

$$\begin{aligned} \lambda^{i+1} &= \alpha(\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top)^{-1}(\mathbf{A}(-\mathbf{G}^{-1}\mathbf{F} - \mathbf{G}^{-1}\mathbf{A}^\top\lambda^i) - \mathbf{b}) \\ &\quad + \lambda^i + \beta(\lambda^i - \lambda^{i-1}) \\ &= -(\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top)^{-1}\mathbf{b} - (\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top)^{-1}\mathbf{A}\mathbf{G}^{-1}\mathbf{F}, \end{aligned} \tag{12}$$

in which λ^{i+1} is independent of λ^i, meaning that the algorithm is dead-beat. We will show below that in presence of constraints the algorithm still converges.

Remark 2. Only in case $\Phi_\omega = \mathbf{G}^{-1}$ is diagonal, the projection becomes a clipping/saturation operation. In case of non-diagonal $\Phi_\omega = \mathbf{G}^{-1}$, efficient algorithms exist for solving the weighted projection in (9), see, e.g., [16]. Alternatively, one could explore sequential quadratic programming algorithms with (diagonally) approximated Hessian matrices.

B. Convergence Analysis

In this section, we will prove the convergence of method (11) for $\beta = 0$. Later, this will be used to propose an algorithm that resets β to zero whenever the algorithm tends to diverge.

The analysis will be based on the observation that (11) can be rewritten as a dynamic system that admits a Lur'e structure, i.e., a linear/affine system with states λ with a static nonlinearity. To show this, we rewrite (11) for $\beta = 0$ as

$$\begin{cases} \lambda^{i+1} &= \lambda^i + \mathcal{B}v^i + \mathcal{E} \\ z^i &= \mathcal{C}\lambda^i + \mathcal{F} \\ v^i &= \max\{\underline{\omega}, \min\{\overline{\omega}, z^i\}\}, \end{cases} \tag{13}$$

with matrices

$$\begin{aligned} \mathcal{B} &= \alpha(\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top)^{-1}\mathbf{A}, & \mathcal{E} &= -\alpha(\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top)^{-1}\mathbf{b}, \\ \mathcal{C} &= -\mathbf{G}^{-1}\mathbf{A}^\top, & \mathcal{F} &= -\mathbf{G}^{-1}\mathbf{F}. \end{aligned} \tag{14}$$

The static nonlinearity is depicted in Figure 1. Because of the element-wise saturation/clipping of the nonlinearity, we can bound the nonlinearity as

$$v_\ell(v_\ell - z_\ell) \leqslant 0, \tag{15}$$

where ℓ denotes the ℓ-th element of v and z. This bound on the nonlinearity will be used in the stability analysis presented below, in which we use a Lyapunov function $V(\hat{\lambda} - \hat{\lambda}^\star) = (\lambda - \lambda^\star)^\top\mathcal{P}(\lambda - \lambda^\star)$, for some well-chosen positive definite matrix $\mathcal{P}$, the full-block S-procedure [17] with some well-chosen scaling, and LaSalle's invariance principle.

Theorem 3. *Assume that the optimal control problem is feasible and that an optimal solution $(\omega^\star, \lambda^\star)$ exists. Given $\beta = 0$, the equilibrium point $\lambda^\star$ of* (13) *is asymptotically stable if $\alpha \in (0, 1]$.*

Proof. Since we assume the problem is feasible, there exist $(\lambda^\star, v^\star, z^\star)$ that satisfy

$$\begin{cases} \lambda^\star &= \lambda^\star + \mathcal{B}v^\star + \mathcal{E} \\ z^\star &= \mathcal{C}\lambda^\star + \mathcal{F} \\ v^\star &= z^\star, \end{cases} \tag{16}$$

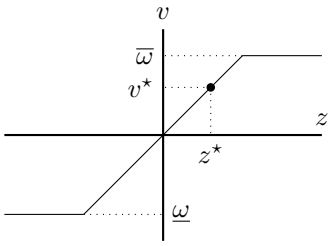

Fig. 1. The nonlinear relation between z and v as saturation function, where ($z^\star$,$v^\star$) is a critical point within the bounds.

meaning that we write

$$\begin{cases} \lambda^{i+1} - \lambda^\star &= (\lambda^i - \lambda^\star) + \mathcal{B}(v^i - v^\star) \\ z^i - z^\star &= \mathcal{C}(\lambda^i - \lambda^\star) \\ v^i - v^\star &= \max\{\underline{\omega}, \min\{\overline{\omega}, z^i\} - z^\star. \end{cases} \tag{17}$$

Because (15) holds element-wise we need to establish that

$$V(\lambda^{i+1}-\lambda^\star)-V(\lambda^i-\lambda^\star) \leqslant (v^i-v^\star)^\top \mathcal{S}(v^i-v^\star-z^i+z^\star), \tag{18}$$

for some diagonal positive-definite matrix $\mathcal{S}$ and this ensures that the Lyapunov function is non-increasing. Substituting (17) into this expression leads to

$$\begin{bmatrix} \lambda^i-\lambda^\star \\ v^i-v^\star \end{bmatrix}^\top \begin{bmatrix} 0 & \mathcal{P}\mathcal{B} + \frac{1}{2}\mathcal{C}^\top\mathcal{S} \\ \mathcal{B}^\top\mathcal{P} + \frac{1}{2}\mathcal{S}\mathcal{C} & \mathcal{B}^\top\mathcal{P}\mathcal{B} - \mathcal{S} \end{bmatrix} \begin{bmatrix} \lambda^i-\lambda^\star \\ v^i-v^\star \end{bmatrix} \leqslant 0. \tag{19}$$

Now, by choosing $\mathcal{P} = \mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top$ and $\mathcal{S} = 2\alpha\mathbf{G}^{-1}$, the northeast (and southwest) corner of the matrix in (19) become zero. To satisfy (19), we need that

$$\mathcal{B}^\top\mathcal{P}\mathcal{B} - \mathcal{S} = \alpha^2\mathbf{A}^\top(\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top)^{-1}\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top(\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top)^{-1}\mathbf{A} - 2\alpha\mathbf{G}^{-1} \preceq 0. \tag{20}$$

By applying the Schur complement twice, this is implied by

$$(1 - \tfrac{\alpha^2}{2\alpha})\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top \succeq 0, \tag{21}$$

holds for all $\alpha \in (0, 1]$.

To show that $V(\lambda - \lambda^\star)$ is a decreasing Lyapunonv function along solution of (17) and for all $\lambda \neq \lambda^\star$, we proceed using a LaSalle's invariance argument. Namely, in the case that $\omega \notin [\underline{\omega}, \overline{\omega}]$, (15) holds strictly for some elements of v and z, meaning that the Lyapunonv function decreases. For the case that all $\omega \in [\underline{\omega}, \overline{\omega}]$, which means that $v = z$, we have that

$$\begin{cases} \lambda^{i+1} - \lambda^\star &= (\lambda^i - \lambda^\star) + \mathcal{B}(z^i - z^\star) \\ z^i - z^\star &= \mathcal{C}(\lambda^i - \lambda^\star), \end{cases} \tag{22}$$

which means that

$$\lambda^{i+1}-\lambda^\star = \lambda^i-\lambda^\star+\mathcal{B}\mathcal{C}(\lambda^i-\lambda^\star) = (1-\alpha)(\lambda^i-\lambda^\star), \tag{23}$$

since $\mathcal{B}\mathcal{C} = \alpha$. This means that $V(\lambda^{i+1} - \lambda^\star) - V(\lambda^i - \lambda^\star) = 0$ is only positive invariant for $\lambda^i = \lambda^\star$ provided that $\alpha \in (0, 1]$. This establishes that the equilibrium point $\lambda^\star$ of (13) is asymptotically stable, which completes the proof. □

In case constraints are absent, or inactive, $\alpha = 1$ delivers the solution in the least amount of iterations. If constraints are present and active, the robustness with respect to the non-linearity may be poor, yet not destabilizing. Therefore, $0 < \alpha < 1$ might lead to better convergence than $\alpha = 1$ in presence of constraints.

IV. Implementation Details

To complete the algorithm introduced in Section III, this section introduces the adaptive restart of the parameter β, the termination criterion used for the algorithm and some details on how the final algorithm can be implemented in MATLAB.

A. Adaptive Restart

The process of tuning α and β can be very tedious, where for $\beta \neq 0$ a convergence proof has not been established. Therefore, a different scheme is introduced in this section. A time-varying β is considered where at each step the value is increased until 1 is reached. One such variable step-size is given in [18], where β_i is chosen to be

$$\beta_i = \frac{i}{i+b}, \tag{24}$$

for $i \in \mathbb{N}$ and $b > 0$ is used to tune the speed at which β sweeps from zero to one.

A disadvantage of accelerating gradient schemes is that it causes side-effects such as high momentum. This causes the algorithm to miss crucial points, as seen in [14]. In order to avoid this phenomenon, the gradient scheme from the same paper is implemented, such that β_i is reset to zero when the Lagrangian of the preconditioned system starts to decrease (as we aim at maximizing this Lagrangian over λ), which equals to

$$\nabla_\lambda L(\lambda^{i-1})^\top(\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top)^{-\top}(\lambda^i - \lambda^{i-1}) \leqslant 0, \tag{25}$$

where L is the Lagrangian given in (7). For the actual implementation, the scheme is rewritten by introducing $\hat{\lambda}^i = \lambda^i + \beta(\lambda^i - \lambda^{i-1})$, and using the update rule for λ as in (11), allowing us to rewrite (25) as

$$(\lambda^i - \hat{\lambda}^{i-1})^\top(\lambda^i - \lambda^{i-1}) \leqslant 0. \tag{26}$$

The advantage of this notation is that the vectors in this notation are available already, whereas calculation the value of the gradient of the Lagrangian would add extra unnecessary steps. This restart rule has also been shown to work with the Fista algorithm [19]. Although convergence for the inclusion of the heavy-ball has not been proven, the reset rule adds robustness to the algorithm. We will show in the next section that this approach leads to a reduction in required iterations.

B. Termination Criteria

The problem is solved as soon as $\omega^\star$ and $\lambda^\star$ satisfy (8). Due to the choice of the preconditioning, the first condition in (8) always hold. Therefore, we consider the solution to be of sufficient accuracy as soon as

$$||\mathbf{A}\omega^{i+1} - \mathbf{b}||_2 \leqslant \varepsilon, \tag{27}$$

where ε is the tolerance of the algorithm.

C. Solving Sparse Matrices

The efficiency of the algorithm is dictated by the speed at which the inverse of $\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top$ can be calculated. A disadvantage of solving this inverse explicitly is that sparsity is lost. A direct method for solving sparse linear systems is recommended. MATLAB implements this using the '\' operator in an extremely efficient manner.

A simple implementation in MATLAB pseudo-code is given in Algorithm 1. In total, this example was coded with less than 30 lines of code, which emphasizes the simplicity of the algorithm.

Algorithm 1 MATLAB pseudo-code

Initialize:
  Select α and b and a desired accuracy ε.
  Compute $\Phi_1 = \mathbf{G}^{-1}\mathbf{A}^\top$, $\Phi_2 = \mathbf{G}^{-1}\mathbf{F}$,
  $\Phi_3 = \mathbf{A}\mathbf{G}^{-1}\mathbf{A}^\top$,
  Set $\lambda = -\Phi_3 \backslash (\mathbf{b} + \mathbf{A}\mathbf{G}^{-1}\mathbf{F})$, $\lambda^- = \mathbf{0}$,
  $i = 1$
loop
  $\omega \leftarrow \max\{\underline{\omega}, \min\{\overline{\omega}, -\Phi_1\lambda - \Phi_2\}\}$
  if $\|A\omega - b\|_2 \leqslant \varepsilon$ **then**
    return λ, ω
  end if
  $\hat{\lambda} \leftarrow \lambda + \frac{i}{i+b}(\lambda - \lambda^-)$
  $\lambda^- \leftarrow \lambda$
  $\lambda \leftarrow \hat{\lambda} + \alpha\,\Phi_3 \backslash (A\omega - b)$
  if $(\lambda - \hat{\lambda})^\top(\lambda - \lambda^-) \leqslant 0$ **then**
    $i \leftarrow 0$
  else
    $i \leftarrow i + 1$
  end if
end loop

V. Numerical Examples

In this section, the computational performance of the proposed heavy-ball projected primal-dual method (HBPPDM) is demonstrated by solving the OCP for two examples. In particular, we consider an example of an inverted pendulum that is subject to a time-varying state constraint and an example of a reference tracking problem of an ATFI-F16 aircraft, as considered in [13]. As the proposed algorithm requires two tuning parameters to be chosen, namely α and b, as in (24), we will first investigate the influence of these parameters on the number of iterations the algorithm requires to converge. Subsequently, we will compare the computation times of our algorithm with quadprog, CPLEX and OSQP for the two examples for various horizons K, where it should be noted that the HBPPDM proposed in this paper is coded in MATLAB, while CPLEX and OSQP both use better optimized high-level coding languages for their algorithms. All our comparisons have been performed on a desktop computer equipped with an Intel i7-3770 processor and 16Gb of system memory, and using $\varepsilon = 10^{-4}$ for the tolerance on primal feasibility.

Example 1. *The inverted pendulum is given in continuous time by $\dot{x} = A_c x + B_c u$ with*

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.048 & 1.58 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.080 & 18.97 & 0 \end{bmatrix}, B_c = \begin{bmatrix} 0 & 0 \\ 0.96 & 1.61 \\ 0 & 0 \\ 1.61 & 0.96 \end{bmatrix}, \quad (28)$$

and is discretized with zero-order hold with a sampling time of 0.1s. We take $Q = I$ and $R = 0.1I$ in (1a)*. Furthermore, the constraints are $\overline{u} = -\underline{u} = [4, 4]^\top$, $\overline{x} = [10, \hat{x}, 5, 10]^\top$, $\underline{x} = [-10, -5, -10, -5]^\top$. $\hat{x} = 3\sin(\frac{2\pi k}{9} + 0.5\pi) + 1$ to ensure that significant control effort is required over time.*

Example 2. *The ATFI-F16 model from [13] is given in continuous time by $\dot{x} = A_c x + B_c u$ with*

$$A_c = \begin{bmatrix} -0.0151 & -60.5651 & 0 & -32.1740 \\ -0.0001 & -1.3411 & 0.9929 & 0 \\ 0.0002 & 43.2541 & -0.8694 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (29)$$

$$B_c = \begin{bmatrix} -2.516 & -0.1689 & -17.251 & 0 \\ -13.136 & -0.2514 & -1.5766 & 0 \end{bmatrix}^\top,$$

and is discretized with zero-order hold with a sampling time of 0.05s. The cost matrices are given by $Q = \mathrm{diag}(10^{-4}, 10^2, 10^{-3}, 10^2)$, and $R = 2 \cdot 10^{-2} I$. The constraints are taken as $\overline{x} = -\underline{x} = [10^6, 0.5, 10^6, 100]^\top$, and $\overline{u} = -\underline{u} = [25, 25]^\top$. A step reference is requested in the fourth state, i.e., $r_k = [0, 0, 0, \hat{r}_k]^\top$ with $\hat{r}_k = 10$ for $1 \leqslant k < K/2$ and zero elsewhere.

A. Tuning

As indicated in Algorithm 1, the HBPPD requires three parameters to be chosen, namely, the tolerance on primal feasibility ε, the step size α and the parameter b that controls the speed at which the momentum increases from zero to one.

Since we selected $\varepsilon = 10^{-4}$, we investigate the influence of α and b on the number of iterations the algorithm requires to satisfy (26). We do this by computing this number for a large number of possible α and b for $K = 1000$ for the two examples under consideration. The results are shown in Fig. 2. It can be seen in this figure that for both examples, the minimum number of iterations is achieved around $\alpha = 0.51$, while there is a less obvious trend in b. We therefore select $b = 28$ in the remainder of this section. This trend is also observed for other horizons lengths, which suggest that tuning is fairly robust and that these parameters are a 'good' choice.

B. Comparison Against Other Solvers

We will now compare the HBPPD method with quadprog, CPLEX and OSQP in terms of CPU time needed to solve the OCP for increasing horizon length K. We compare the CPU time needed to solve the optimization problem (2) (and not the time needed to build the matrices $\mathbf{G}$, $\mathbf{F}$, $\mathbf{A}$ and $\mathbf{b}$, which is the same for all solvers). The results are shown in Fig. 3. It can be seen that the HBPPD method outperforms CPLEX for Example 1, while the computational

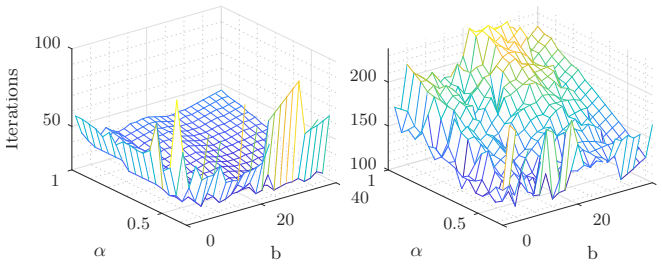

Fig. 2. Iterations needed to achieve convergence as function of α and b for Example 1 (left) and Example 2 (right) for $K = 1000$. Cases with iterations exceeding 100 have not been rendered.

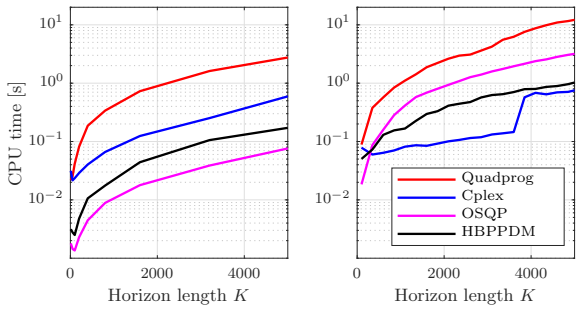

Fig. 3. CPU time as function of horizon length using different solvers for Example 1 (left) and Example 2 (right).

performance is slightly worse than OSQP. For Example 2, the computational performance of the HBPPD method is better than OSQP and only slightly worse than CPLEX for larger problems.

Example 2 also shows that the HBPPD method is capable of finding accurate solutions in a case where OSQP cannot. As can be seen in the left figure of Fig. 4, in which we compare the achieved cost of all solvers with quadprog, we can observe that quadprog, CPLEX and HBPPD achieve the same cost, while OSQP shows a different cost. Furthermore, the accuracy at which $\|\mathbf{A}\omega - \mathbf{b}\|$ is achieved is different for all solvers. As can be seen in the right figure of Fig. 4, both OSQP and HBPPD are less accurate than quadprog and CPLEX. For OSQP the limited accuracy causes the optimal cost to deviate from CPLEX and quadprog, while for the novel HBPPD method is the limited accuracy does not noticeably impact the optimal cost. The limited accuracy does enable achieving fast computation times for the HBPPD method even though the implementation is simple and can possibly be further optimized.

VI. Conclusion

In this paper, we have proposed an heavy-ball accelerated projected primal-dual algorithm for solving large-horizon linear quadratic optimal control problems. We have formulated the optimal control problem as a sparse static optimization problem and shown that using a simple splitting of the KKT conditions an algorithm can be derived which is flexible and has a small code size. Furthermore, we have shown that by writing the method as a Lur'e system leads to proving that

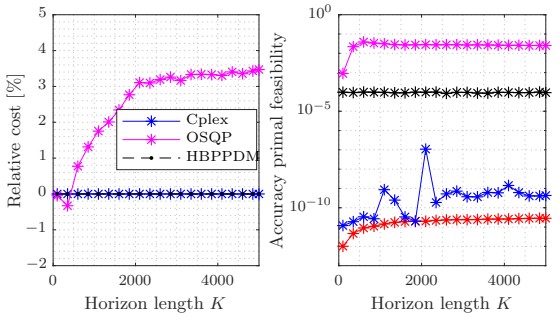

Fig. 4. Relative cost with respect to quadprog (left) and achieved accuracy of the primal feasibility (right) for different solvers as function of horizon length.

the primal-dual algorithm converges. In order to ease the tuning of algorithm and speed up convergence, a variable step-size and restart rule for α has been proposed in order to slow down the algorithm if the solution tends to diverge. The numerical analysis demonstrates that the algorithm is very capable of delivering fast computational times and challenges current state-of-the-art solvers.

References

[1] A. G. Beccuti, G. Papafotiou, and M. Morari, "Optimal control of the boost dc-dc converter," in *Proc IEEE CDC*, 2005.

[2] R. Salehi, A. Stefanopoulou, S. Mahesh, and M. Allain, "Reduced-order long-horizon predictive thermal management for diesel engine aftertreatment systems," in *Proc American Control Conf*, 2019.

[3] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta numerica*, 1995.

[4] S. Boyd and J. Mattingley, "Branch and bound methods," *Notes for EE364b, Stanford University*, 2007.

[5] J. M. Maciejowski, *Predictive Control with Constraints*. Pearson, 2001.

[6] IBM Corp: IBM ILOG CPLEX V12.1, *User's Manual for CPLEX (2009)*.

[7] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Math. Prog. Comput.*, 2020.

[8] D. Liao-McPherson and I. Kolmanovsky, "FBstab: A proximally stabilized semismooth algorithm for convex quadratic programming," *Automatica*, 2020.

[9] A. Lenoir and P. Mahey, "A survey on operator splitting and decomposition of convex programs," *RAIRO - Operations Research*, 2017.

[10] E. K. Ryu and S. Boyd, "A primer on monotone operator methods," *Appl. Comput. Math.*, 2016.

[11] B. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Comput. Math. and Math. Physics*, 1964.

[12] Y. Nesterov, "Accelerating the cubic regularization of Newton's method on convex problems," *Mathematical Programming*, 2008.

[13] P. Giselsson and S. Boyd, "Metric selection in fast dual forward–backward splitting," *Automatica*, 2015.

[14] B. O'Donoghue and E. Candès, "Adaptive restart for accelerated gradient schemes," *Found. of Comput. Math.*, 2013.

[15] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides, "A condensed and sparse QP formulation for predictive control," in *Proc IEEE CDC*, 2011.

[16] S. Becker, J. Fadili, and P. Ochs, "On quasi-Newton forward-backward splitting: Proximal calculus and convergence," *SIAM Journal on Optimization*, 2019.

[17] C. Scherer, *Robust mixed control and LPV control with full block scalings*, ser. Advances in Design & Control. Springer-Verlag, 1999.

[18] G. Stathopoulos, M. Korda, and C. N. Jones, "Solving the infinite-horizon constrained LQR problem using splitting techniques," *IFAC Proc.*, 2014.

[19] T. Goldstein, C. Studer, and R. G. Baraniuk, "A field guide to forward-backward splitting with a FASTA implementation," *CoRR*, 2014.