Block Structured Preconditioning within an Active-Set Method for Real-Time Optimal Control

Quirynen, R.; Knyazev, A.; Di Cairano, S.

TR2018-081 July 12, 2018

Abstract

Model predictive control (MPC) requires solving a block-structured optimal control problem at each sampling instant. We propose an iterative preconditioned solver with computational cost that scales linearly with the number of intervals and quadratically with the number of state and control variables, and can be efficiently implemented on embedded hardware for real-time optimal control. Block-structured factorizations and low-rank updates are combined with blockdiagonal preconditioning within a primal active-set strategy (PRESAS). Multiple numerical tests using our preliminary C implementation demonstrate competitiveness with the state-of-the-art, as illustrated on an ARM Cortex-A53 processor.

European Control Conference (ECC)

Block Structured Preconditioning within an Active-Set Method for Real-Time Optimal Control

Rien Quirynen[1], Andrew Knyazev[1], Stefano Di Cairano[1]

***Abstract*— Model predictive control (MPC) requires solving a block-structured optimal control problem at each sampling instant. We propose an iterative preconditioned solver with computational cost that scales linearly with the number of intervals and quadratically with the number of state and control variables, and can be efficiently implemented on embedded hardware for real-time optimal control. Block-structured factorizations and low-rank updates are combined with block-diagonal preconditioning within a primal active-set strategy (PRESAS). Multiple numerical tests using our preliminary C implementation demonstrate competitiveness with the state-of-the-art, as illustrated on an ARM Cortex-A53 processor.**

I. INTRODUCTION

We are interested in solving the following discrete time formulation of a convex constrained linear-quadratic optimal control problem (OCP)

$$\min_{X,U} \quad \sum_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} x_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} Q_k & S_k^\top \\ S_k & R_k \end{bmatrix} \begin{bmatrix} x_k \\ u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^\top \begin{bmatrix} x_k \\ u_k \end{bmatrix} \tag{1a}$$

$$+ \frac{1}{2} x_N^\top Q_N x_N + q_N^\top x_N \tag{1b}$$

$$\text{s.t.} \quad x_0 = \hat{x}_0, \tag{1c}$$

$$x_{k+1} = a_k + A_k x_k + B_k u_k, \ k = 0, \ldots, N-1, \tag{1d}$$

$$0 \geq d_k + D_k^{\mathrm{x}} x_k + D_k^{\mathrm{u}} u_k, \ k = 0, \ldots, N, \tag{1e}$$

where we define the state vectors as $x_k \in \mathbb{R}^{n_\mathrm{x}}$, the control inputs as $u_k \in \mathbb{R}^{n_\mathrm{u}}$ and the cost matrices as $Q_k \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$, $S_k \in \mathbb{R}^{n_\mathrm{u} \times n_\mathrm{x}}$ and $R_k \in \mathbb{R}^{n_\mathrm{u} \times n_\mathrm{u}}$. The constraints include the system dynamics with $A_k \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$, $B_k \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{u}}$, the inequality constraints with $D_k^{\mathrm{x}} \in \mathbb{R}^{n_{\mathrm{c},k} \times n_\mathrm{x}}$, $D_k^{\mathrm{u}} \in \mathbb{R}^{n_{\mathrm{c},k} \times n_\mathrm{u}}$ and an initial value condition where $\hat{x}_0 \in \mathbb{R}^{n_\mathrm{x}}$ denotes the current state estimate. The quadratic program (QP) in (1) exhibits the sparsity structure, as it typically arises in a linear or linear time-varying formulation of model predictive control (MPC) [1]. A similarly structured QP arises as a subproblem within a sequential quadratic programming (SQP) method for solving nonlinear optimal control problems [2].

The following requirements are important to be taken into account when designing or choosing an embedded QP solver for industrial applications of real-time optimal control:

1) scaling of computational complexity and memory requirements with problem dimensions N, n_x and n_u,
2) warm starting capabilities for receding horizon control,
3) portability of solver code and dependencies,
4) deterministic or early termination of solver in real-time applications to obtain feasible but suboptimal solution,
5) numerical performance on embedded control hardware with limited computational resources,
6) reliability and ease of understanding, implementing and maintaining by non-experts.

Most embedded optimization algorithms that have successfully been applied to real-time optimal control rely on direct linear algebra routines. However, it is known that iterative methods can result in a better asymptotic complexity when solving the saddle point linear systems arising in second order optimization methods [3]. Iterative solvers, such as the minimal residual (MINRES) method [4], are suitable for hardware acceleration of the linear system solution, e.g., using an FPGA for fast embedded applications [5], due to their higher ratio of addition and multiplication operations. The use of an iterative method also allows us to make a trade-off between computational time and numerical accuracy of the solution as in [6]. However, for a general saddle point linear system, Krylov subspace methods tend to converge poorly without preconditioning [3].

The prior work in [7], [8] already studied a particular block-diagonal preconditioner within an inexact interior-point (IP) framework. However, the linear systems within an IP method become increasingly ill-conditioned as the solution is approached [8], [9]. Instead, we propose two alternative block-diagonal preconditioners within an active-set strategy. Unlike IP methods, an active-set solver can considerably benefit from the use of warm- or hot-starting techniques to reduce the computational effort when solving a sequence of closely related optimal control problems as discussed in [2], [10], [11]. In addition, the cost per iteration is generally of a lower computational complexity by exploiting low-rank updates of the matrix factorizations when changing the current guess for the active set [12], [13]. The proposed `PRESAS` solver allows for an initial setup computational complexity of $\mathcal{O}(Nm^3)$ and a per-iteration complexity of $\mathcal{O}(Nm^2)$ for the QP in (1), where m denotes the number of state and control variables in the system. The setup computational cost is incurred only once per QP solution and it can be performed offline when using hot-starting within real-time optimal control.

The paper is organized as follows. Section II presents the preliminaries on quadratic programming and on active-set methods. The techniques for preconditioning of iterative solvers are presented in Section III. The particular block structure exploitation for real-time optimal control is presented in Section IV. Section V discusses the implementation of the `PRESAS` algorithm and its computational performance is illustrated in Section VI based on numerical case studies.

[1]Mitsubishi Electric Research Laboratories, Cambridge, MA, 02139, USA. `quirynen@merl.com`

II. PRELIMINARIES

We assume that the convex QP in (1) has a unique global solution that is non-degenerate. A solution is degenerate when either the strict complementarity condition or the linear independence constraint qualification (LICQ) does not hold [14]. In order to have a unique solution to the QP in (1), the Hessian needs to be positive definite on the null-space of the strictly active constraints in the solution.

A. Embedded Optimal Control Algorithms

Generally, there is a trade-off between solvers that make use of second-order information and require only few but computationally complex iterations, e.g., `qpOASES` [12], versus first-order methods that are of low complexity but may require many more iterations, such as `PQP` [15], `ADMM` [16] and other gradient or splitting-based methods [2]. In addition, there is an important distinction between optimal control algorithms that target the dense versus the sparse problem formulation. The numerical elimination of the state variables in a condensing routine [17] is generally of a computational complexity $\mathcal{O}(N^2 m^3)$, but it can be mostly avoided in linear MPC applications. However, even with an offline preparation of the dense QP formulation, solvers applied to this dense QP will have a runtime complexity of $\mathcal{O}(N^2 m^2)$ [13]. Instead, we focus on directly solving the OCP formulation with the block sparsity structure in (1), similar to the software tools in `FORCES` [18] and `HPMPC` [19].

It is important to note that many tailored QP algorithms for real-time optimal control rely on strict convexity of the cost function. This enables the usage of a dual Newton strategy such as in `qpDUNES` [20], sparsity exploiting linear algebra routines such as the block-tridiagonal Cholesky factorization of the Schur complement in [21], [22] or a particular Riccati recursion for linear-quadratic control problems in [11], [23]. In the case of a positive semidefinite cost matrix, regularization needs to be applied, followed by an iterative refinement procedure to obtain a solution to the original problem. This combination of regularization and iterative refinement can also be needed in the presence of badly conditioned QP matrices. Instead, we do not require strict convexity and employ an iterative method to solve each linear KKT system.

B. Primal Feasible Active-Set Method

The basic idea behind active-set methods is to find the optimal active set by iteratively updating a current guess. When fixing the active constraints at the current solution guess, a corresponding structured equality constrained QP needs to be solved to compute a new search direction

$$\min_{\Delta X, \Delta U} \quad \sum_{k=0}^{N-1} \frac{1}{2} \Delta w_k^\top H_k \, \Delta w_k + \begin{bmatrix} \tilde{q}_k^\top & \tilde{r}_k^\top \end{bmatrix} \Delta w_k \tag{2a}$$

$$+ \frac{1}{2} \Delta x_N^\top Q_N \Delta x_N + \tilde{q}_N^\top \Delta x_N \tag{2b}$$

$$\text{s.t.} \quad \Delta x_0 = 0, \tag{2c}$$

$$\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k, \; k = 0, \dots, N-1, \tag{2d}$$

$$0 = D^{\mathrm{x}}_{k,i} \Delta x_k + D^{\mathrm{u}}_{k,i} \Delta u_k, \quad (k,i) \in \mathcal{W}, \tag{2e}$$

where $\mathcal{W}$ denotes the current guess for the active set, i.e., the *working set*. The variables $\Delta w_k := (\Delta x_k, \Delta u_k) = (x_k - \bar{x}_k, u_k - \bar{u}_k)$ are defined for $k = 0, \dots, N-1$ and $\Delta w_N := \Delta x_N = x_N - \bar{x}_N$, where $\bar{w}_k := (\bar{x}_k, \bar{u}_k)$ denotes the current guess for the optimal solution of the QP in (1). Note that the equality constrained QP in (2) results in the search direction $\bar{w}_k + \alpha \Delta w_k$ for which all constraints in the set $\mathcal{W}$ remain satisfied, regardless of the value for α. A distinction should be made between primal, dual and primal-dual active-set methods [14]. In addition, parametric methods have been proposed [12] in order to exploit the parametric aspect within a homotopy framework. In this work, we consider a primal feasible active-set method.

III. BLOCK-DIAGONAL PRECONDITIONING

At each iteration of the active-set method, one needs to efficiently solve the saddle point linear system

$$\begin{bmatrix} \mathcal{H} & \mathcal{A}^\top \\ \mathcal{A} & \mathbb{0} \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} h \\ a \end{bmatrix} \quad \text{or} \quad \mathcal{K} z = b, \tag{3}$$

which corresponds to the first order necessary conditions of optimality for the equality constrained QP (2). In (3), the matrix $\mathcal{A}$ has full rank and $\mathcal{H}$ is symmetric and positive semidefinite. Unlike the prior work on embedded optimization algorithms for optimal control based on direct linear algebra routines in [11], [13], [22], [23], we propose the use of iterative solvers as discussed for general saddle point linear systems in [3], [24]. Preconditioning is necessary for the good performance of iterative solvers [6]. It produces a modified linear system $\mathcal{P}^{-1} \mathcal{K} z = \mathcal{P}^{-1} b$ where $\mathcal{P}$ is the preconditioner, which is such that

1) computations with the operator $\mathcal{P}^{-1}$ are cheaper than solving the original saddle point linear system in (3),
2) and the preconditioned matrix $\mathcal{P}^{-1}\mathcal{K}$ approximates the identity or its eigenvalues are tightly clustered [4].

An overview on algebraic and application-specific preconditioners can be found in [3]. Here, we focus on two block-diagonal preconditioning techniques.

A. Block-Diagonal Preconditioners

To obtain a good performance for the iterative solver, we can use one of the standard block-diagonal preconditioners

$$\mathcal{P}_{\mathrm{a}} = \begin{bmatrix} \mathcal{H} + \mathcal{A}^\top \Gamma \mathcal{A} & \mathbb{0} \\ \mathbb{0} & \Gamma^{-1} \end{bmatrix} \quad \text{or} \quad \mathcal{P}_{\mathrm{s}} = \begin{bmatrix} \tilde{\mathcal{H}} & \mathbb{0} \\ \mathbb{0} & \mathcal{A} \tilde{\mathcal{H}}^{-1} \mathcal{A}^\top \end{bmatrix}, \tag{4}$$

where Γ is a symmetric positive definite weighting matrix and $\tilde{\mathcal{H}} \approx \mathcal{H}$ such that $\tilde{\mathcal{H}} \succ 0$. A popular choice for the weighting matrix, which follows an augmented Lagrangian type technique [24], is $\Gamma = \gamma \mathbb{1}$ where $\gamma > 0$ and $\mathbb{1}$ denotes the identity matrix. Note that the inverse of the preconditioning matrices $\mathcal{P}_{\mathrm{a}}^{-1}$ or $\mathcal{P}_{\mathrm{s}}^{-1}$ is indeed easier to apply than solving the original linear system, because of the block-diagonal structure. In addition, the block matrices in (4) are designed to be positive definite, which enables the use of a Cholesky factorization. For the specific case of solving optimal control problems, the block-tridiagonal structure of $\mathcal{A}\tilde{\mathcal{H}}^{-1}\mathcal{A}^\top$ or $\mathcal{H}+\gamma\mathcal{A}^\top\mathcal{A}$ should be exploited. This will be the topic of discussion in Section IV.

For the specific case of $\mathcal{P}_\mathrm{a}$, as discussed in detail by [25], the eigenvalues of the preconditioned matrix become more tightly clustered around ± 1 as the value of $\gamma > 0$ increases. For a sufficiently large value of γ, MINRES can therefore converge within two iterations in the ideal setting. However, choosing γ too large may result in ill-conditioning of the matrix $\mathcal{P}_\mathrm{a}$. On the other hand, when using the Schur-complement based preconditioner $\mathcal{P}_\mathrm{s}$ and $\tilde{\mathcal{H}} = \mathcal{H}$ is invertible, the preconditioned matrix has three distinct eigenvalues 1, $\frac{1}{2}(1+\sqrt{5})$ and $\frac{1}{2}(1-\sqrt{5})$, and therefore MINRES converges within three iterations [26]. The eigenvalues become (tightly) clustered when, e.g., regularization is needed to make $\tilde{\mathcal{H}} \approx \mathcal{H}$ positive definite [3], [26].

B. Preconditioned MINRES Algorithm

Both the Schur-complement based $\mathcal{P}_\mathrm{s}$ and the augmented Lagrangian preconditioner $\mathcal{P}_\mathrm{a}$ are symmetric positive definite, given a sufficiently large choice for $\Gamma = \gamma\,\mathbb{1}$. Therefore, symmetric preconditioning $L^{-1}\mathcal{K}\,L^{-\top}$ can be performed instead of left preconditioning $\mathcal{P}^{-1}\mathcal{K}$, where $\mathcal{P} = LL^\top$. This results in the preconditioned minimal residual (PMINRES) algorithm as described in [4]. PMINRES is based on the three-term recurrence in the Lanczos iteration for symmetric matrices and specifically requires a positive definite preconditioner [3]. On the other hand, the generalized minimal residual (GMRES) algorithm is based on the Arnoldi iteration, for which the computational cost generally grows with each iteration, but it does not need a positive definite preconditioner. For simplicity, we further focus on using PMINRES to solve the symmetric linear system.

IV. OPTIMAL CONTROL STRUCTURED PRECONDITIONERS

The saddle point linear system in (3) describes the first order necessary conditions of optimality for the equality constrained QP in (2). It has a particular sparsity structure because $\mathcal{H}$ corresponds to the block-diagonal Hessian matrix and the constraint matrix $\mathcal{A}$ reads as

$$\mathcal{A} = \begin{bmatrix} -\mathbb{1} & \mathbb{0} & & \\ E_0^\mathrm{x} & E_0^\mathrm{u} & & \\ A_0 & B_0 & -\mathbb{1} & \mathbb{0} \\ & & & \ddots \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} -\mathbb{1} & \mathbb{0} \end{bmatrix} & \\ E_0 & \\ C_0 & \begin{bmatrix} -\mathbb{1} & \mathbb{0} \end{bmatrix} \\ & \ddots \end{bmatrix}, \tag{5}$$

where E_k^x and E_k^u denote the active inequality constraints for each interval k, corresponding to the working set in $\mathcal{W}$. For notational convenience, the block matrices $C_k := \begin{bmatrix} A_k & B_k \end{bmatrix}$ and $E_k := \begin{bmatrix} E_k^\mathrm{x} & E_k^\mathrm{u} \end{bmatrix}$ have been defined.

Any iterative method for solving the linear system in (3) requires the efficient computation of matrix-vector multiplications for the block-structured matrix $\mathcal{K}$. It is straightforward to directly exploit such sparsity structure for each matrix-vector multiplication, resulting in an asymptotic computational complexity of $\mathcal{O}(Nm^2)$. Another computational effort is applying the preconditioning operators $\mathcal{P}_\mathrm{a}^{-1}$ or $\mathcal{P}_\mathrm{s}^{-1}$.

A. Preconditioner I: Augmented Lagrangian

The application of the augmented Lagrangian type preconditioner $\mathcal{P}_\mathrm{a}$ in (4) requires the factorization of the block-tridiagonal matrix $\mathcal{H}+\gamma\mathcal{A}^\top\mathcal{A}$ which reads as

$$\mathcal{H}+\gamma\mathcal{A}^\top\mathcal{A} = \begin{bmatrix} \hat{H}_0 + \gamma\, G_0^\top G_0 & -\gamma\, C_0^\top & \\ -\gamma\, C_0 & \hat{H}_1 + \gamma\, G_1^\top G_1 & -\gamma\, C_1^\top \\ & -\gamma\, C_1 & \ddots \end{bmatrix}, \tag{6}$$

where $G_k = \begin{bmatrix} E_k \\ C_k \end{bmatrix}$ and $\hat{H}_k = \begin{bmatrix} Q_k + \gamma\,\mathbb{1} & S_k^\top \\ S_k & R_k \end{bmatrix}$. This matrix is positive definite for any value $\gamma > 0$ such that a block-tridiagonal Cholesky factorization [21] can be applied. This procedure is based on a sequence of standard dense factorizations for block matrices of dimension $m = n_\mathrm{x} + n_\mathrm{u}$, denoted as $L = \mathtt{chol}(P)$ such that $LL^\top = P$. The computational complexity for this factorization is therefore $\mathcal{O}(Nm^3)$. However, such computational cost is incurred only once per QP solution for the initial guess of active constraints. In addition, as discussed in Section V, it can be completely avoided in the online computations by using a hot-starting procedure for linear MPC.

Rank-one Factorization Updates: It is well known that a Cholesky factorization for a matrix $LL^\top = P \in \mathbb{R}^{n\times n}$ can be recovered in $\mathcal{O}(n^2)$ computations after a rank-one modification $\tilde{P} = P + \alpha vv^\top$. This is referred to as an update if $\alpha > 0$ and as a downdate if $\alpha < 0$. In case of the proposed active-set strategy, a rank-one modification is needed for the matrix $\mathcal{H}+\gamma\mathcal{A}^\top\mathcal{A}$ whenever a constraint is either added or removed from the current working set, i.e., a row is added or removed from the Jacobian matrix $\mathcal{A}$ in (5). By preserving the block-tridiagonal sparsity structure, this update procedure can be carried out with a computational complexity of $\mathcal{O}(Nm^2)$. One could alternatively employ a *backward* or *reverse* variant of the Cholesky factorization, which has particular advantages in the context of MPC applications as discussed in [20].

B. Preconditioner II: Schur Complement

The Schur complement type preconditioner $\mathcal{P}_\mathrm{s}$ in (4) results also in a block-tridiagonal structure

$$\mathcal{A}\tilde{\mathcal{H}}^{-1}\mathcal{A}^\top = \begin{bmatrix} \tilde{Q}_0^{-1} & -\tilde{Q}_0^{-1}{E_0^\mathrm{x}}^\top & -\tilde{Q}_0^{-1}A_0^\top & \\ -E_0^\mathrm{x}\tilde{Q}_0^{-1} & E_0\tilde{H}_0^{-1}E_0^\top & E_0\tilde{H}_0^{-1}C_0^\top & \\ -A_0\tilde{Q}_0^{-1} & C_0\tilde{H}_0^{-1}E_0^\top & C_0\tilde{H}_0^{-1}C_0^\top + \tilde{Q}_1^{-1} & -\tilde{Q}_1^{-1}{E_1^\mathrm{x}}^\top \\ & & -E_1^\mathrm{x}\tilde{Q}_1^{-1} & \ddots \end{bmatrix}, \tag{7}$$

where, for simplicity, we have assumed that each of the Hessian block matrices reads as $H_k := \begin{bmatrix} Q_k & \mathbb{0} \\ \mathbb{0} & R_k \end{bmatrix}$, i.e., $S_k = \mathbb{0}$ for $k = 0, \ldots, N-1$. In the case of (7), note that each of the blocks is of different dimensions, corresponding to the number of active constraints in each block. As mentioned earlier, if the Hessian block matrix H_k is positive semidefinite, then a positive definite approximation $\tilde{H}_k = \begin{bmatrix} \tilde{Q}_k & \mathbb{0} \\ \mathbb{0} & \tilde{R}_k \end{bmatrix} \succ 0$ needs to be computed.

We further restrict to a standard regularization procedure of the form $\tilde{H}_k = H_k + \epsilon \mathbb{1}$. The value for $\epsilon > 0$ needs to be chosen sufficiently small such that $\tilde{H}_k \approx H_k$ but it also needs to be large enough such that $\tilde{H}_k \succ 0$. In practice, it is common for Q_k, R_k to be diagonal such that the value for ϵ can be easily chosen for each of the diagonal elements individually. In addition, such a diagonal structure results in a computationally efficient way to compute each of the block matrices in (7). However, the Hessian matrices do not need to be diagonal and one can even include off-diagonal matrices S_k within H_k for $k = 0, \ldots, N-1$, without changing the block-tridiagonal sparsity structure in (7). For this general setting, one can rely on a Cholesky decomposition for each of the Hessian blocks in order to efficiently compute the products with $\tilde{H}_k^{-1}$ in (7). Whenever a particular Hessian block is not strictly positive definite, one can apply an *on-the-fly* regularization in the form of a modified Cholesky factorization, e.g., as in [14], [20]. Alternatively, we can apply the structure-exploiting regularization technique tailored to optimal control from [27].

Rank-one Factorization Updates: One practical advantage of the augmented Lagrangian preconditioner, that can immediately be observed, is that the symmetric positive definite matrix $\mathcal{H} + \gamma \mathcal{A}^\top \mathcal{A}$ in (6) is of a fixed dimension with $N(n_\mathrm{x} + n_\mathrm{u}) + n_\mathrm{x}$ rows and columns. On the other hand, the alternative Schur complement based preconditioning results in the block-tridiagonal matrix $\mathcal{A}\tilde{\mathcal{H}}^{-1}\mathcal{A}^\top$ with variable dimensions, because the amount of rows and columns corresponds directly to the number of active constraints. The advantage of this is that the matrix $\mathcal{A}\tilde{\mathcal{H}}^{-1}\mathcal{A}^\top$ will always have less than $N(n_\mathrm{x} + n_\mathrm{u}) + n_\mathrm{x}$ rows and columns, because the total number of active constraints needs to be (strictly) less than the amount of variables. Removing a constraint results in a standard rank-one update ($\alpha > 0$) of the block-tridiagonal Cholesky factorization for the resulting smaller matrix. This procedure becomes slightly more involved when adding a constraint to the current working set, resulting in a particular variant of a rank-one downdate ($\alpha < 0$) instead. But in both cases, these updates can be performed with a computational complexity of $\mathcal{O}(Nm^2)$.

V. PRESAS IMPLEMENTATION FOR EMBEDDED PREDICTIVE CONTROL

The proposed `PRESAS` solver relies on the MINRES method in combination with the optimal control structured preconditioners within the primal feasible active-set strategy. A few implementation aspects deserve some more detailed discussion to apply this solver in a real-time embedded context for predictive control or estimation.

A. Finding a Primal Feasible Initial Point

A primal active-set method requires the availability of an initial point, which is already primal feasible. In the general case of a constrained quadratic program, this is a nontrivial task that corresponds to a *Phase I* procedure as described, for example, in [14], [28]. However, when solving the parametric OCP in (1) within receding horizon based control or estimation, it becomes relatively easy to satisfy the initial value condition and the continuity constraints based on a forward simulation using a shifted version of the previous control trajectory. Given an MPC design that offers recursive feasibility, the resulting solution guess can be made primal feasible under nominal conditions. In practice, one should introduce slack variables in order to always be able to satisfy all state-dependent inequality constraints in (1e). The latter approach is quite standard in practical MPC implementations in order to ensure a feasible solution to the QP at each sampling instant. As discussed also in [14], an L1 penalty can be used with sufficiently high weights in order to guarantee that a primal feasible solution is found whenever possible.

B. Warm-starting for Model Predictive Control

In embedded applications, where a sequence of closely related optimization problems needs to be solved, *warm-starting* of the algorithm can considerably reduce the corresponding computational effort. This procedure is relatively straightforward in active-set methods, where an initial solution guess and a corresponding working set are needed. This is an advantage of active-set algorithms over interior-point methods, for which warm-starting typically plays a much less important role [2], [10]. Particularly for `PRESAS`, only one matrix factorization of the block-structured preconditioner is needed for each QP solution. In a linear time-invariant MPC implementation, one can perform the block-tridiagonal Cholesky factorization offline and limit the online computations to rank-one updates instead. Note that this requires a possibly multi-rank update of the Cholesky factors from the solution of one QP to the solver initialization for the subsequent QP. This is typically referred to as *hot-starting*, as implemented also in `qpOASES` [12].

C. Algorithmic Advantages and Disadvantages

The active-set identification suffers from a combinatorial complexity and is therefore *NP hard* in the worst case [10]. However, active-set methods have been successfully used in many real-time control applications [2]. Specifically for `PRESAS`, a primal feasible but suboptimal solution is always obtained when terminating the algorithm before convergence is reached. In addition, the proposed solver adheres to all other embedded optimal control requirements (1-6) that were mentioned in the introduction. Namely, `PRESAS` enjoys the preferred computational complexity of $\mathcal{O}(Nm^2)$ per iteration based on its block sparsity structure exploitation and the use of iterative linear algebra routines. For being an active-set method with tailored optimal control type structure exploitation, the proposed approach is considerably easier to implement than prior work such as [13]. The next section illustrates the numerical performance of `PRESAS` based on a self-contained solver implementation in C code, running on an ARM Cortex-A53 that is relatively close to an embedded microprocessor with limited computational resources.

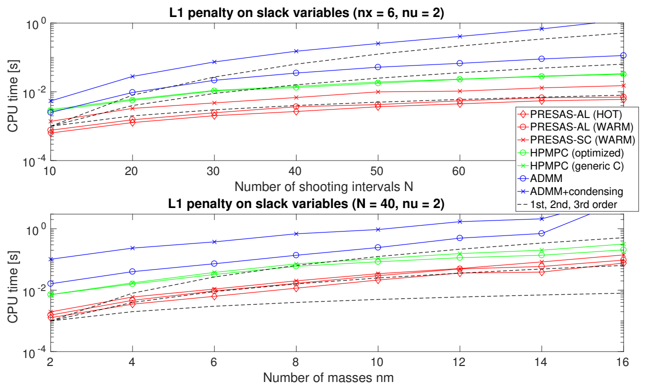

Fig. 1. Average computation times for MPC on the chain of masses with L1 slack reformulation: varying number of intervals N and masses n_{m}.

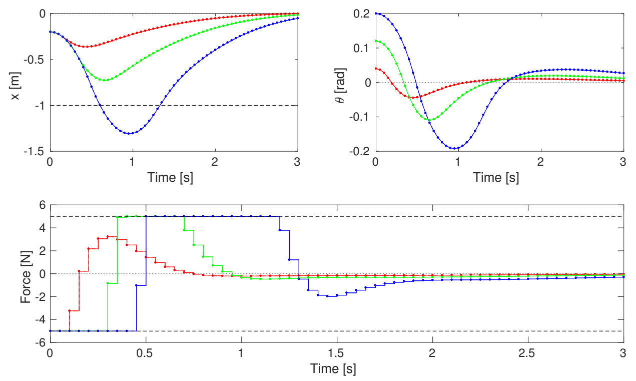

Fig. 2. Closed-loop trajectories for linear MPC to stabilize an inverted pendulum, starting from different initial conditions for the angle θ.

VI. NUMERICAL CASE STUDIES

We consider two fundamentally different case studies. The first case study, featuring a chain of spring-connected masses as in [22], includes constraints on all states and controls of which only relatively few constraints become active in a closed-loop simulation. The second example, on the other hand, involves a linearized model of the inverted pendulum case study in [29] with only a few constraints, resulting in a relatively large number of active-set changes. The presented simulation results show that the proposed PRESAS solver, based on a primal active-set method, can generally compete with state of the art optimal control algorithms for both case studies. Note that this is illustrated based on a plain C code implementation, which is standalone and therefore does not rely on any advanced linear algebra packages in order to be easily embedded. All computation times have been obtained using an ARM Cortex-A53 processor in the Raspberry Pi 3.[1]

A. Case Study 1: Chain of Oscillating Masses

The linear time-invariant system dynamics and corresponding OCP formulation for the chain of oscillating masses, of the form in (1), are described in [22]. The full state of the system consists of the displacement and velocity of the n_{m} masses, i.e., $x(t) \in \mathbb{R}^{2n_{\mathrm{m}}}$ such that the state dimension can be varied by changing the amount of masses. A number of actuators $n_{\mathrm{u}} < n_{\mathrm{m}}$ apply tensions between certain masses while respecting the actuator limitations as well as constraints on the position and velocity of each of the masses. In order to guarantee the QP to remain feasible at each sampling instant, a slack variable is introduced for the state constraints on each shooting interval. Based on an L1 type penalization of this additional variable in the objective, a feasible solution can be found whenever possible.

Figure 1 shows the average computation times per QP solution during the closed-loop MPC simulations for different numbers of masses n_{m} and for a varying control horizon length N. The considered variant of ADMM in [16] solves the small but dense QP after numerically eliminating the state variables in a condensing routine. Figure 1 shows the computation time for ADMM, either including or not including the time needed for condensing. HPMPC is an interior-point method that directly solves the optimal control structured QP in (1), similar to the proposed PRESAS algorithm. For completeness, we included the computational results both using the hardware tailored and the general-purpose implementation of HPMPC [19]. It can be observed from Figure 1 that PRESAS-AL and PRESAS-SC, respectively using the block preconditioner from Section IV-A and IV-B, can outperform the other solvers on this particular case study. This performance seems to scale well with the number of shooting intervals and the number of state variables. Figure 1 also shows how hot-starting can lead to an additional speedup by eliminating the need for the block-tridiagonal Cholesky factorization of the preconditioner.

B. Case Study 2: Control of an Inverted Pendulum

This second numerical case study involves stabilizing the nonlinear inverted pendulum in the upward unstable position, based on a linearization of the nonlinear system dynamics in that steady state [29]. The closed-loop trajectories are illustrated in Figure 2 for different initial conditions for the angle of the pendulum $\theta_0 = 0.12$, 0.16 or 0.20. Note that both the actuated force and the cart position are constrained to remain within their respective bounds. As illustrated in Figure 2, the position constraint is violated in case of the initial value $\theta_0 = 0.20$, which is again treated based on a slack reformulation of the state constraints. This results in a relatively high number of online active-set changes with respect to the amount of state $n_{\mathrm{x}} = 4$ and control variables $n_{\mathrm{u}} = 2$. Table I shows the corresponding average computation times for linear MPC of the inverted pendulum, including the solvers ADMM, HPMPC and PRESAS, but also based on PQP, qpOASES and qpDUNES.

Table I shows the detailed computation times and numbers of solver iterations for the closed-loop MPC simulations on an ARM Cortex-A53 with double-precision arithmetics. Note that warm-starting of the solvers is used, except for the interior point method in HPMPC. For most of the QP solvers, the computational effort grows significantly when increasing the initial value for the angle of the pendulum.

[1]The Raspberry Pi 3 uses a Broadcom BCM2837 SoC with a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor, with 512 KB shared L2 cache.

TABLE I

MPC COMPUTATION TIMES (MS) FOR INVERTED PENDULUM WITH VARYING INITIAL CONDITION ($T_\mathrm{s} = 50$ MS AND $N = 50$).

	$\theta_0 = 0.12$		$\theta_0 = 0.16$		$\theta_0 = 0.20$	
	time [ms] (mean/max)	# iter (mean/max)	time [ms] (mean/max)	# iter (mean/max)	time [ms] (mean/max)	# iter (mean/max)
`PQP`	12.1/207.8	2/51	156.7/2345.1	50/761	857.9/3071.3	282/1000
`ADMM`	1.12/20.98	1.9/12	3.94/40.87	12.8/83	8.09/35.55	33.5/154
`qpOASES`	3.93/10.54	0.1/2	6.94/33.63	1.4/12	22.27/80.05	9.7/41
`qpDUNES`	2.06/9.45	2.1/4	3.14/16.98	2.5/7	4.94/28.79	3.0/10
`HPMPC`	**4.43/9.34**	4.8/7	**5.39/11.63**	5.8/10	**5.93/12.03**	7.0/10
`PRESAS-SC`	1.06/2.95	1.1/4	1.39/18.56	1.4/25	3.63/44.05	4.3/58
`PRESAS-AL`	0.96/2.55	1.1/4	1.28/17.32	1.4/25	3.18/42.97	4.3/58
`PRESAS-SC` (15)	**1.06/2.95**	1.1/4	**1.34/12.36**	1.4/15	**3.06/13.58**	3.5/15
`PRESAS-AL` (15)	**0.96/2.55**	1.1/4	**1.29/9.80**	1.4/15	**2.60/10.65**	3.5/15

However, by using the primal active-set method in `PRESAS`, a feasible but suboptimal solution can always be obtained by limiting the maximum number of active-set changes. In this particular case study, `PRESAS` with a maximum of 15 active-set changes shows the same trajectories as in Figure 2 because of the feedback action of MPC. In addition, by setting such a maximum number of online active-set changes, the algorithm remains competitive in terms of its worst-case computational performance as illustrated in Table I.

VII. CONCLUSIONS

This paper proposed two different block-structured preconditioning techniques within a primal active-set method for fast real-time optimal control applications. More specifically, an augmented Lagrangian type and a Schur complement based block preconditioner are used in the minimal residual (MINRES) method. In addition, factorization update techniques are proposed to effectively embed the block-structured preconditioned residual method within a primal active-set strategy (PRESAS). Based on two numerical case studies of linear MPC, its computational performance is illustrated for a proof of concept implementation in C code on an ARM Cortex-A53 processor.

REFERENCES

[1] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill, 2017.

[2] H. J. Ferreau, S. Almer, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, J. L. Jerez, G. Stathopoulos, and C. Jones, "Embedded optimization methods for industrial automatic control," in *Proc. IFAC World Congress*, 2017.

[3] M. Benzi, G. H. Golub, and J. Liesen, "Numerical solution of saddle point problems," *Acta Numerica*, pp. 1–137, 2005.

[4] A. Greenbaum, *Iterative Methods for Solving Linear Systems*. Society for Industrial and Applied Mathematics, 1997.

[5] D. Boland and G. A. Constantinides, "An FPGA-based implementation of the MINRES algorithm," in *2008 International Conference on Field Programmable Logic and Applications*, Sept 2008, pp. 379–384.

[6] A. Knyazev, Y. Fujii, and A. Malyshev, "Preconditioned continuation model predictive control," in *Proc. IEEE Conf. on Control Appl.*, 2015.

[7] A. Shahzad, E. Kerrigan, and G. Constantinides, "Preconditioners for inexact interior point methods for predictive control," in *Proc. American Control Conf.*, 2010, pp. 5714–5719.

[8] A. Shahzad, E. C. Kerrigan, and G. A. Constantinides, "A fast well-conditioned interior point method for predictive control," in *Proc. IEEE Conf. on Decision and Control*, Dec 2010, pp. 508–513.

[9] T. Rees and C. Greif, "A preconditioner for linear systems arising from interior point optimization methods," *SIAM Journal on Scientific Computing*, vol. 29, no. 5, pp. 1992–2007, 2007.

[10] R. Bartlett, A. Wächter, and L. Biegler, "Active Set vs. Interior Point Strategies for Model Predictive Control," in *Proc. American Control Conf.*, Chicago, Il, 2000, pp. 4229–4233.

[11] S. Wright, "Applying new optimization algorithms to model predictive control," in *5th Int. Conf. on Chem. Proc. Control*, 1996, pp. 147–155.

[12] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl, "qpOASES: a parametric active-set algorithm for quadratic programming," *Math. Prog. Comp.*, vol. 6, no. 4, pp. 327–363, 2014.

[13] C. Kirches, H. G. Bock, J. P. Schlöder, and S. Sager, "A factorization with update procedures for a KKT matrix arising in direct optimal control," *Math. Prog. Comp.*, vol. 3, no. 4, pp. 319–348, 2011.

[14] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[15] S. Di Cairano, M. Brand, and S. A. Bortoff, "Projection-free parallel quadratic programming for linear model predictive control," *International Journal of Control*, vol. 86, no. 8, pp. 1367–1385, 2013.

[16] A. U. Raghunathan and S. Di Cairano, "ADMM for convex quadratic programs: Q-linear convergence and infeasibility detection," *arXiv:1411.7288*, 2015.

[17] H. G. Bock and K. J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," in *Proc. IFAC World Congress*. Pergamon Press, 1984, pp. 242–247.

[18] A. Domahidi and J. Perez, "FORCES professional," embotech GmbH (http://embotech.com/FORCES-Pro), July 2014.

[19] G. Frison, H. B. Sorensen, B. Dammann, and J. B. Jørgensen, "High-performance small-scale solvers for linear model predictive control," in *Proc. European Control Conf.*, June 2014, pp. 128–133.

[20] J. V. Frasch, S. Sager, and M. Diehl, "A parallel quadratic programming method for dynamic optimization problems," *Mathematical Programming Computations*, vol. 7, no. 3, pp. 289–329, 2015.

[21] E. Anderson, Z. B. C., Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*. Philadelphia: SIAM, 1999.

[22] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[23] G. Frison and J. B. Jørgensen, "Efficient implementation of the Riccati recursion for solving linear-quadratic control problems," in *Proc. IEEE Conf. on Control Appl.*, Aug 2013, pp. 1117–1122.

[24] M. Benzi and A. J. Wathen, *Some Preconditioning Techniques for Saddle Point Problems*. Springer, 2008, pp. 195–211.

[25] C. Greif and D. Schoetzau, "Preconditioners for saddle point linear systems with highly singular (1,1) blocks," *Electronic transactions on numerical analysis*, vol. 22, pp. 114–121, 2006.

[26] M. F. Murphy, G. H. Golub, and A. J. Wathen, "A note on preconditioning for indefinite linear systems," *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 1969–1972, 2000.

[27] R. Verschueren, M. Zanon, R. Quirynen, and M. Diehl, "A sparsity preserving convexification procedure for indefinite quadratic programs arising in direct optimal control," *SIAM Journal of Optimization*, vol. 27, no. 3, pp. 2085–2109, 2017.

[28] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. Chichester: Wiley, 1987.

[29] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators," *Opt. Control Appl. & Meth.*, vol. 36, pp. 685–704, 2014.