QPALM-OCP: A Newton-Type Proximal Augmented Lagrangian Solver Tailored for Quadratic Programs Arising in Model Predictive Control

Kristoffer Fink Løwenstein, Daniele Bernardini, and Panagiotis Patrinos

***Abstract*—In model predictive control fast and reliable quadratic programming solvers are of fundamental importance. The inherent structure of the subsequent optimal control problems can lead to substantial performance improvements if exploited. Therefore, we present a structure-exploiting solver based on proximal augmented Lagrangian, extending the general-purpose quadratic programming solver `QPALM`. Our solver relies on semismooth Newton iterations to solve the inner sub-problem while directly accounting for the optimal control problem structure via efficient and sparse matrix factorizations. The matrices to be factorized depend on the active-set and therefore low-rank factorization updates can be employed like in active-set methods resulting in cheap iterates. We compare our solver with `QPALM` and other well-known solvers and show its benefits in a numerical example.**

***Index Terms*—Optimization algorithms, optimal control, predictive control for nonlinear systems.**

I. Introduction

MODEL Predictive Control (MPC) relies on online optimization to repeatedly compute the solution of constrained Optimal Control Problems (OCPs) in a receding horizon fashion. In linear MPC the online optimization problem is inherently a Quadratic Program (QP) and in Nonlinear Model Predictive Control (NLMPC) the solution procedure is often based on Sequential Quadratic Programming (SQP) relying on the solutions of quadratic approximations of the original nonlinear problem. Therefore, efficient and reliable QP solvers are of paramount importance for the use of MPC in industrial (and, especially, embedded) applications, as they typically must adhere to strict real-time requirements. Recently, the QP solver `QPALM`, based on a proximal Augmented Lagrangian Method (ALM) [1], has been proposed and shown to be very reliable and efficient [2], [3]. `QPALM` solves a series of unconstrained minimization problems, exploiting fast smooth optimization techniques, in which the linear system solved in the iterates only changes with the active-set. Therefore, `QPALM` can benefit from low-rank factorization updates as active-set methods and iterations are therefore cheaper than in Interior Point Methods (IPMs). In combination with a closed-form exact line search, this yields a very fast minimization strategy. Contrary to traditional active-set methods, it allows for substantial changes in active-set in each iteration speeding up the convergence. When implementing efficient MPC methods, tailored solvers that can exploit the problem structure can provide significant speed-ups compared to general-purpose solvers, thus extending the applicability of MPC to challenging real-time problems. The contribution of this letter is a QP solver tailored for solving optimization problems arising in MPC. The proposed method is an extension of the general-purpose solver `QPALM`, that explicitly accounts for the equality constraints arising from the system dynamics and exploits the problem structure when solving the now equality-constrained sub-problem of the ALM iterates. The proposed method preserves low-rank updates of the matrix factorizations, exploiting the problem structure, and significantly benefits from warm-starting. The remainder of this letter is structured as follows. In Section II we state the OCP problem. In Section III we introduce the equality-constrained `QPALM`. Section IV introduces the structure exploiting semismooth Newton method solving the ALM-subproblem. Numerical results are presented in Section V and conclusions are drawn in Section VI.

Notation: With $\mathbf{Sym}(\mathbb{R}^n)$ we indicate the set of $\mathbb{R}^{n\times n}$ symmetric matrices, whereas $\mathbf{Sym}_+(\mathbb{R}^n)$ and $\mathbf{Sym}_{++}(\mathbb{R}^n)$ denote the subsets which are positive semidefinite and positive definite, respectively. With $\mathbf{1}_n \in \mathbb{R}^n$ we denote a vector containing ones. Given a nonempty closed convex set $C \subseteq \mathbb{R}^n$, with $\Pi_C(x)$ we indicate the projection of a point x onto C, namely $\Pi_C(x) = \arg\min_{y\in C} \|y - x\|$. $\mathbf{dist}_\Sigma(x, C)$ denotes the distance from x to set C in the Euclidean norm induced by $\Sigma \in \mathbf{Sym}_{++}(\mathbb{R}^n)$.

Manuscript received 8 March 2024; revised 5 May 2024; accepted 21 May 2024. Date of publication 6 June 2024; date of current version 26 June 2024. This work was supported by the European Union's 2020 Research and Innovation Programme under the Marie Skłodowska-Curie (ELO-X) under Agreement 953348. Recommended by Senior Editor S. Olaru. *(Corresponding author: Kristoffer Fink Løwenstein.)*

Kristoffer Fink Løwenstein is with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy, and also with ODYS S.r.l., 20159 Milano, Italy (e-mail: kristoffer.fink.loewenstein@polimi.it).

Daniele Bernardini is with ODYS S.r.l., 20159 Milano, Italy (e-mail: daniele.bernardini@odys.it).

Panagiotis Patrinos is with the Department of Electrical Engineering (ESAT-STADIUS), KU Leuven, 3001 Leuven, Belgium (e-mail: panos.patrinos@kuleuven.be).

Digital Object Identifier 10.1109/LCSYS.2024.3410638

II. Problem Formulation

We consider the OCP defined by

$$\begin{aligned}
\underset{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}}{\textbf{minimize}} \quad & \sum_{j=0}^{N-1} \ell_j(x_j,u_j) + \ell_N(x_N) && \text{(1a)}\\
\text{s.t.} \quad & x_0 = \bar{x}_0 && \text{(1b)}\\
& x_{j+1} = A_j x_j + B_j u_j + e_j, \ \forall j = 0\cdots N-1 && \text{(1c)}\\
& \underline{c}_j \le C_j x_j + D_j u_j \le \bar{c}_j, \quad \forall j = 0\cdots N-1 && \text{(1d)}\\
& \underline{c}_N \le C_N x_N \le \bar{c}_N && \text{(1e)}
\end{aligned}$$

with stage cost

$$\ell_j(x_j,u_j) \ := \ \tfrac{1}{2}\begin{bmatrix} x_j \\ u_j \end{bmatrix}^T \begin{bmatrix} Q_j & S_j^T \\ S_j & R_j \end{bmatrix} \begin{bmatrix} x_j \\ u_j \end{bmatrix} + \begin{bmatrix} q_j \\ r_j \end{bmatrix}^T \begin{bmatrix} x_j \\ u_j \end{bmatrix} \quad \text{and}$$

terminal cost $\ell_N(x_N) := \frac{1}{2}x_N^T Q_N x_N + q_N^T x_N$. Further, $\bar{x}_0 \in \mathbb{R}^{n_x}$ is the initial state, $x_j \in \mathbb{R}^{n_x}$ and $u_j \in \mathbb{R}^{n_u}$ are the state and control variables for the j-th stage, and $N \in \mathbb{N}$ is the prediction horizon. The LTV system dynamics are defined by $A_j \in \mathbb{R}^{n_x \times n_x}$, $B_j \in \mathbb{R}^{n_x \times n_u}$, and $e_j \in \mathbb{R}^{n_x}$, potentially as a result of linearizing nonlinear dynamics. The stage-wise costs on the states and control inputs are defined by $Q_j \in \mathbf{Sym}_+(\mathbb{R}^{n_x})$, $R_j \in \mathbf{Sym}_+(\mathbb{R}^{n_u})$, $q_j \in \mathbb{R}^{n_x}$, and $r_j \in \mathbb{R}^{n_u}$, whereas a mixed quadratic cost is defined through $S_j \in \mathbb{R}^{n_x \times n_u}$. Further, we consider mixed state and input bounds for $j = 0,\dots,N-1$ defined by $\underline{c}_j \in \mathbb{R}^{n_{c_j}}$, $\bar{c}_j \in \mathbb{R}^{n_{c_j}}$ and the matrices $C_j \in \mathbb{R}^{n_{c_j} \times n_x}$ and $D_j \in \mathbb{R}^{n_{c_j} \times n_u}$, where n_{c_j} denotes the number of box constraints for the j-th stage. Finally, the terminal constraint is defined by $\underline{c}_N \in \mathbb{R}^{n_{cN}}$, $\bar{c}_N \in \mathbb{R}^{n_{cN}}$ and the matrix $C_N \in \mathbb{R}^{n_{cN} \times n_x}$. The OCP in (1) can be written as a generic quadratic program

$$\begin{aligned}
\underset{x\in\mathbb{R}^n}{\textbf{minimize}} \quad & f(x) && \text{(2a)}\\
\textbf{subject to} \quad & Mx = b && \text{(2b)}\\
& l_b \le Ax \le u_b && \text{(2c)}
\end{aligned}$$

where states and inputs are collected in vector $x \in \mathbb{R}^n$ defining the primal variables of (2) with $n = (N+1)n_x + Nn_u$. The objective is $f(x) = \frac{1}{2}x^T Q x + q^T x$ with $Q \in \mathbb{R}^{n\times n}$ and $q \in \mathbb{R}^n$. The constraints defining the initial state (1b) and the system dynamics (1c) are collected in $M \in \mathbb{R}^{p\times n}$ and $b \in \mathbb{R}^p$ where $p = (N+1)n_x$. Lastly, the bounds on states and inputs (1d)-(1e) are collected in $A \in \mathbb{R}^{m\times n}$ and $l_b, u_b \in \mathbb{R}^m$ with $m = \sum_{j=0}^{N} n_{c_j}$. Next, we extend the QPALM algorithm proposed in [3] to explicitly cope with the equality constraints arising in OCPs.

III. Equality Constrained QPALM

Consider again the generic QP in (2) where the auxiliary variable $z \in \mathbb{R}^m$ is introduced

$$\begin{aligned}
\underset{x\in\mathbb{R}^n, z\in\mathbb{R}^m}{\textbf{minimize}} \quad & f(x) && \text{(3a)}\\
\textbf{subject to} \quad & Mx = b && \text{(3b)}\\
& l_b \le z \le u_b && \text{(3c)}\\
& Ax = z. && \text{(3d)}
\end{aligned}$$

QPALM solves (3) by minimizing the proximal Σ_y-augmented Lagrangian

$$\mathcal{L}_{\Sigma_y}(x,z,y) := f(x) + y^T(Ax - z) + \frac{1}{2}\|Ax - z\|^2_{\Sigma_{y,k}} + \frac{1}{2}\|x - \hat{x}^k\|_{\Sigma^{-1}_{x,k}} \quad (4)$$

defined by relaxing (3d) and introducing the Lagrange multipliers $y \in \mathbb{R}^m$, the corresponding diagonal weight matrix $\Sigma_{y,k} \in \mathbf{Sym}_{++}(\mathbb{R}^m)$ and the proximal term $\|x - \hat{x}^k\|_{\Sigma^{-1}_{x,k}}$ with corresponding weight matrix $\Sigma_{x,k} \in \mathbf{Sym}_{++}(\mathbb{R}^n)$ (for details, see [3]). The proximal term is important when Q and R in (1) are only positive semidefinite or if solving a non-convex QP sub-problem in nonlinear MPC. For fixed Lagrange multipliers, y^k, the update of the primal and auxiliary variables amounts to minimizing (4) with respect to x and z:

$$\begin{aligned}
\left(x^{k+1}, z^{k+1}\right) = \ & \underset{x\in\mathbb{R}^n, z\in\mathbb{R}^m}{\operatorname{argmin}} \ \mathcal{L}_{\Sigma_{y,k}}\left(x,z,y^k\right) && \text{(5a)}\\
& \text{s.t. } Mx = b && \text{(5b)}\\
& l_b \le z \le u_b && \text{(5c)}
\end{aligned}$$

where the index k refers to the k-th ALM iteration. The (x,z)-update in (5) can be decomposed into two separate steps. Consider again (4), but adding and combining with the constant term $\frac{1}{2}\|y^k\|_{\Sigma_{y,k}}$ yields $\mathcal{L}_{\Sigma_y}(x,z,y^k) = f(x) + \frac{1}{2}\|Ax - z + \Sigma^{-1}_{y,k}y^k\|^2_{\Sigma_{y,k}} + \frac{1}{2}\|x - \hat{x}^k\|_{\Sigma^{-1}_{x,k}}$. Therefore, minimizing (5) first with respect to z results in

$$z^{k+1} = \underset{z\in C}{\operatorname{argmin}} \frac{1}{2}\|Ax - z + \Sigma^{-1}_{y,k}y^k\|_{\Sigma_{y,k}} = \Pi_C\left(Ax + \Sigma^{-1}_{y,k}y^k\right) \quad (6)$$

where $C := \{z \in \mathbb{R}^m \mid l_b \le z \le u_b\}$. The second equality in (6) is due to C being seperable and $\Sigma_{y,k}$ diagonal. Inserting (6) in (5) yields a problem defined only in the primal variables:

$$\begin{aligned}
x^{k+1} = \ & \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} \ \phi_k(x) && \text{(7a)}\\
& \text{s.t. } Mx = b && \text{(7b)}
\end{aligned}$$

where

$$\phi_k(x) := f(x) + \frac{1}{2}\mathbf{dist}^2_{\Sigma_{y,k}}\left(Ax + \Sigma^{-1}_{y,k}y^k, C\right) + \frac{1}{2}\|x - \hat{x}^k\|_{\Sigma^{-1}_{x,k}} \quad (8)$$

Therefore, one ALM iteration of QPALM amounts to three subsequent steps: (*i*) update the primal variables by solving (7), (*ii*) update the auxiliary variables by the projection in (6), and (*iii*) update the dual variables using a standard ALM approach [4]. Clearly, the main computational burden is related to finding the solution of (7) which is solved using an iterative method due to the piecewise quadratic structure caused by the squared distance. Different from the subproblem in [3], the inner problem (7) is equality constrained (instead of unconstrained), but can still be efficiently solved using a semismooth Newton method. The equality-constrained QPALM is outlined in Algorithm 1, where i denotes the iterate of the subproblem minimization in (7). The convergence proof of Alg. 1 for nonconvex QPs follows [3, Th. 2.6] closely. We briefly outline the termination criteria, however, for a more detailed description and additional hyperparameters we refer the interested reader to the paper introducing QPALM [3]. Given the absolute and relative tolerance $\epsilon_a\ \epsilon_r > 0$, $(x^{k+1}, y^{k+1}, z^{k+1}, \lambda^{k+1})$ is considered an optimal solution to the problem in (2) if the following termination criteria defined by the first order optimality conditions of (2) are met:

$$\|Qx^{k+1} + q + A^T y^{k+1} + M^T \lambda^{k+1}\|_\infty \le \epsilon_a$$

Algorithm 1 Equality-Constrained QPALM

Inputs: x^0, y^0, λ^0
Outputs: $x^\star, y^\star, \lambda^\star$
Initialize:
Set $\Sigma_{y,0}$ according to [3, Section V, Equation (5.2)];
for $k = 0, 1, \ldots$ **do**
  $x^{k,0} \leftarrow x^k$. Let ϕ_k be as in (8);
  for $i = 0, 1, \ldots$ **do**
    Let (d, λ) be the semismooth Newton direction and τ the corresponding step size (See Sec. IV).
    $x^{k,i+1} = x^{k,i} + \tau d$
    $\lambda^{k,i+1} = (1-\tau)\lambda^{k,i} + \tau\lambda$
    if $(x^{k,i+1}, \lambda^{k,i+1})$ satisfies (10) **then**
      Break;
    end if
  end for
  $x^{k+1} = x^{k,i+1}$
  $\lambda^{k+1} = \lambda^{k,i+1}$
  $z^{k+1} = \Pi_C(Ax^{k+1} + \Sigma_{y,k}^{-1} y^k)$
  $y^{k+1} = y^k + \Sigma_{y,k}(Ax^{k+1} - z^{k+1})$
  if $(x^{k+1}, z^{k+1}, y^{k+1}, \lambda^{k+1})$ satisfies (9) **then**
    Return $(x^\star, y^\star, \lambda^\star) \leftarrow (x^{k+1}, y^{k+1}, \lambda^{k+1})$;
  end if
  Update $\Sigma_y, \Sigma_x, \hat{x}, \delta_a, \delta_r$ according to [3, Section 5];
end for

$$+ \epsilon_r \mathbf{max}\Big(\|Qx^{k+1}\|_\infty, \|q\|_\infty, \|A^T y^{k+1}\|_\infty, \|M^T\lambda^{k+1}\|_\infty\Big) \quad (9a)$$

$$\|Ax^{k+1} - z^{k+1}\|_\infty \le \epsilon_a + \epsilon_r \mathbf{max}\Big(\|Ax^{k+1}\|_\infty, \|z^{k+1}\|_\infty\Big). \quad (9b)$$

The QPALM algorithm converges even if the inner problem is solved inexactly [2]. Hence, the inner problem is initially solved to a coarse tolerance which is gradually decreased according to [3, Sec. 5]. Therefore, the inner loop is terminated when the pair $(x^{k,i+1}, \lambda^{k,i+1})$ satisfies

$$\|\nabla\phi_k\Big(x^{k,i+1}\Big) + M^T\lambda^{k,i+1} + \Sigma_{x,k}^{-1}\Big(x^{k,i+1} - \hat{x}^k\Big)\|_\infty \le \delta_a^k + \delta_r^k \mathbf{max}\Big(\|Qx^{k,i+1}\|_\infty, \|q\|_\infty, \|A^T y^{k,i+1}\|_\infty, \|M^T\lambda^{k,i+1}\|_\infty\Big). \quad (10)$$

Next, we show how to compute the semismooth Newton step efficiently exploiting the OCP structure.

IV. Efficient Structure-Exploiting Semismooth Newton Method

The gradient of $\phi_k(x)$ in (8) is given by (See [3, Sec. 3.1])

$$\nabla\phi_k(x) = \nabla f(x) + A^T\Big(y^k + \Sigma_{y,k}(Ax - \Pi_C(Ax + \Sigma_{y,k}^{-1}y^k))\Big) + \Sigma_{x,k}^{-1}\Big(x - \hat{x}^k\Big). \quad (11)$$

Due to the projection operator, the gradient is not continuously differentiable. However, the generalized Jacobian [5, Sec. 7.1] of Π_C at $Ax + \Sigma_{y,k}^{-1}y^k$ (see, e.g., [6, Sec. 6.2.d]) can be used for which one element is given by the diagonal matrix $P_k(x)$

$$(P_k(x))_{ii} = \begin{cases} 1 \text{ if } l_{b,i} \le \Big(Ax + \Sigma_{y,k}^{-1}y^k\Big)_i \le u_{b,i} \\ 0 \text{ otherwise.} \end{cases} \quad (12)$$

with i indicating the i-th constraint of (3c). Therefore, one element, $H_k(x) \in \mathbf{Sym}_{++}(\mathbb{R}^n)$, of the generalized Hessian of $\phi_k(x)$ is

$$H_k(x) = \nabla^2 f(x) + A^T\Sigma_{y,k}(I - P(x))A + \Sigma_{x,k}^{-1} \quad (13)$$

Denoting the active-set

$$\mathcal{J}(x) := \{i \mid \Big(Ax + \Sigma_{y,k}^{-1}y^k\Big)_i \notin \big[l_{b,i}, u_{b,i}\big]\} \quad (14)$$

one has that $(P(x))_{ii}$ is 1 if $i \in \mathcal{J}(x)$ and 0 otherwise, and thus the generalized Hessian can be written in the more economical form

$$H_k(x) = Q + A_{\mathcal{J}\cdot}^T\big(\Sigma_{y,k}\big)_{\mathcal{J}\mathcal{J}}A_{\mathcal{J}\cdot} + \Sigma_{x,k}^{-1} \quad (15)$$

where $A_{\mathcal{J}\cdot}$ collects all j-th rows of A, with $j \in \mathcal{J}(x)$. Similarly $(\Sigma_{y,k})_{\mathcal{J}\mathcal{J}}$ is obtained by collecting all j-th rows and columns from $\Sigma_{y,k}$ with $j \in \mathcal{J}(x)$.

The semismooth Newton direction satisfies

$$\begin{bmatrix} H_k(x) & M^T \\ M & 0 \end{bmatrix}\begin{bmatrix} d \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla\phi_k(x) \\ -(Mx - b) \end{bmatrix} \quad (16)$$

where $\lambda \in \mathbb{R}^p$ is the Langrange multipliers of the equality constraints. The linear system in (16) collects the first order optimality conditions (KKT) stemming from a quadratic approximation of the problem in (7) and thus resembles the KKT matrix of a general equality constrained QP for which several well-established solution methods exist [4, Ch. 16.1]. One method, namely factorizing the full KKT matrix, must rely on specialized methods, e.g., modified LDL-factorization using carefully selected pivoting blocks [7], due to indefiniteness. We use the Schur-complement method as it provides a way to exploit the structure of the OCP and allows us to work with positive definite matrices and thus obtain numerically stable Cholesky factorizations. Manipulating (16) leads to three separate linear systems

$$H_k(x)\ v = \nabla\phi_k(x) \quad (17a)$$

$$\Psi\lambda = -\big(w - r_{eq}\big) \quad (17b)$$

$$H_k(x)\ d = -s \quad (17c)$$

with $\Psi := MH_k^{-1}(x)M^T \in \mathbf{Sym}_{++}(\mathbb{R}^p)$, $r_{eq} := Mx - b \in \mathbb{R}^p$, $w := Mv \in \mathbb{R}^p$, and $s := (M^T\lambda + \nabla\phi_k(x)) \in \mathbb{R}^n$. The linear system in (17a) is solved stage-wise by exploiting the block diagonal structure of the generalized Hessian. For the j-th stage the generalized Hessian is

$$H_{k,j}(x) = \begin{bmatrix} \tilde{Q}_j & \tilde{S}_j^T \\ \tilde{S}_j & \tilde{R}_j \end{bmatrix} + \Sigma_{x,k,j}^{-1} \in \mathbf{Sym}_{++}\big(\mathbb{R}^{n_x+n_u}\big) \quad (18)$$

where $\tilde{Q}_j = Q_j + C_{j,\mathcal{J}_j\cdot}^T(\Sigma_{y,k,j})_{\mathcal{J}_j\mathcal{J}_j}C_{j,\mathcal{J}_j\cdot}$, $\tilde{R}_j = R_j + D_{j,\mathcal{J}_j\cdot}^T(\Sigma_{y,k,j})_{\mathcal{J}_j\mathcal{J}_j}D_{j,\mathcal{J}_j\cdot}$, and $\tilde{S}_j = S_j + D_{j,\mathcal{J}_j\cdot}^T(\Sigma_{y,k,j})_{\mathcal{J}_j\mathcal{J}_j}C_{j,\mathcal{J}_j\cdot}$ in which the active-set in (14) and the corresponding penalty weights are extracted stage-wise, i.e., using the bounds defined in (1d). The matrix $\Sigma_{x,k,j}$ is the submatrix of $\Sigma_{x,k}$ related to the j-th stage. For the terminal state the generalized Hessian is defined as $H_{k,N}(x) = Q_N + C_{N,\mathcal{J}_N\cdot}^T(\Sigma_{y,k,N})_{\mathcal{J}_N\mathcal{J}_N}C_{N,\mathcal{J}_N\cdot} + \Sigma_{x,k,N}^{-1}$ where $H_{k,N}(x) \in \mathbf{Sym}_{++}(\mathbb{R}^{n_x})$. The solution of (17a) is summarized in Algorithm 2. More specifically, the linear systems at lines 3 and 5 are solved by constructing the Cholesky factorization $H_{k,j}(x) = L_jL_j^T$ of the stage-wise generalized Hessian. Thereby, the order of complexity of the

Algorithm 2 Solving for v and Building w

Inputs: $A_j, B_j, \nabla\phi_k(x), H_{k,j}(x)$
Outputs: w
Solve $H_{k,x_N}(x)v_{x_N} = \nabla\phi_{k,x_N}(x)$;
for $j = N-1, \dots, 0$ **do**:
  Solve $H_{k,j}(x)v_j = \nabla\phi_{k,j}(x)$;
  $w_j = A_j v_{x_j} + B_j v_{u_j} - v_{x_{j+1}}$;
end for
$w_{\bar{x}_0} = -v_{x_0}$

facotorization procedure reduces from $\mathcal{O}(N^3(n_x+n_u)^3)$ to $\mathcal{O}(N(n_x+n_u)^3)$. In addition, the construction of w is done exploiting the structure of M given by (1c). The next step is solving (17b), which is the most expensive step as it requires the factorization of the Schur complement matrix. It has a block-banded structure and therefore specialized methods for block tridiagonal matrices can be used (related to Riccati recursions [8]). The Schur complement Ψ has the form

$$\Psi := \begin{bmatrix} \Psi_{00} & \Psi_{01} & 0 & \dots & 0 \\ \Psi_{01}^T & \Psi_{11} & \Psi_{12} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \Psi_{N-2,N-1}^T & \Psi_{N-1,N-1} & \Psi_{N-1,N} \\ 0 & 0 & 0 & \Psi_{N-1,N}^T & \Psi_{N,N} \end{bmatrix} \tag{19}$$

with entries

$$\Psi_{00} = P_0 H_{k,0}^{-1}(x) P_0^T \in \mathbf{Sym}_{++}\left(\mathbb{R}^{n_x}\right) \tag{20a}$$

$$\Psi_{j,j} = F_{j-1} H_{k,j-1}^{-1}(x) F_{j-1}^T + P_j H_{k,j}^{-1}(x) P_j^T \in \mathbf{Sym}_{++}\left(\mathbb{R}^{n_x}\right) \tag{20b}$$

$$\Psi_{j,j+1} = P_j H_{k,j}^{-1}(x) F_j^T \in \mathbb{R}^{n_x \times n_x} \tag{20c}$$

where the matrices $F_j = \begin{bmatrix} A_j & B_j \end{bmatrix} \in \mathbb{R}^{n_x \times (n_x+n_u)}$ and $P_j = \begin{bmatrix} -\mathbf{I}_{\mathbf{n_x}} & \mathbf{0}_{n_x \times n_u} \end{bmatrix} \in \mathbb{R}^{n_x \times (n_x+n_u)}$ (except $P_N = -\mathbf{I}_{\mathbf{n_x}}$) are sub-matrices of M. Further, we can use the existing Cholesky factorizations of the stage-wise blocks of the generalized Hessian, used to solve (17a) to construct the entries of (20b) and (20c) by forming the auxiliary linear systems

$$V_j L_j^T = F_j \tag{21a}$$

$$W_j L_j^T = P_j \tag{21b}$$

where $V_j, W_j \in \mathbb{R}^{n_x \times (n_x+n_u)}$ which leads to

$$F_{j-1} H_{k,j-1}^{-1}(x) F_{j-1}^T = V_{j-1,j-1} V_{j-1,j-1}^T \tag{22a}$$

$$P_j H_{k,j}^{-1}(x) P_j^T = W_j W_j^T \tag{22b}$$

$$P_j H_{k,j}^{-1}(x) F_j^T = W_j V_j^T. \tag{22c}$$

With the construction of the elements of Ψ, its Cholesky factorization

$$L^\Psi = \begin{bmatrix} L_{00}^\Psi & 0 & 0 & \dots & 0 \\ L_{10}^\Psi & L_{11}^\Psi & 0 & \dots & 0 \\ 0 & L_{21}^\Psi & L_{22}^\Psi & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & L_{N-1,N}^\Psi & L_{N,N}^\Psi \end{bmatrix} \tag{23}$$

Algorithm 3 Block-Triangular Factorization of Ψ

Inputs: $A_j, B_j, H_{k,j}(x) = L_j L_j^T$
Outputs: L^Ψ
Compute L_{00}^Ψ (20a) and W_0 (21b);
for $j = 0, \dots, N-1$ **do**:
  Solve $V_j L_j^T = F_j$ and $W_{j+1} L_{j+1}^T = P_{j+1}$ (21);
  Construct $\Psi_{j,j+1}$ and $\Psi_{j+1,j+1}$ using (20) and (22);
  Solve $\Psi_{j,j+1} = L_{j,j}^\Psi L_{j+1,i}^{\Psi T}$ to get $L_{j+1,i}^\Psi$ (24a);
  Compute $L_{j+1,j+1}^\Psi = \mathrm{chol}(\Psi_{j+1,j+1} - L_{j+1,j}^\Psi L_{j+1,j}^{\Psi T})$ (24b);
end for

Algorithm 4 Solving for d

Inputs: $A_j, B_j, \nabla\phi_k(x), H_k(x), \lambda$
Outputs: d
$s_N = \nabla\phi_{k,x_N}(x) - \lambda_N$;
Solve $H_{k,x_N}(x) d_{x_N} = -s_N$;
for $j = N-1, \dots, 0$ **do**:
  $s_{u_j} = B_j^T \lambda_j + \nabla\phi_{k,u_j}(x)$
  $s_{x_{j+1}} = A_{j+1}^T \lambda_{j+1} - \lambda_j + \nabla\phi_{k,x_{j+1}}(x)$
  Solve $H_{k,j}(x) d_j = -s_j$;
end for

can be computed by utilizing the relations

$$\Psi_{j,j+1} = L_{j,j}^\Psi L_{j+1,j}^{\Psi T} \tag{24a}$$

$$\Psi_{j,j} - L_{j,j-1}^\Psi L_{j,j-1}^{\Psi T} = L_{j,j}^\Psi L_{j,j}^{\Psi T} \tag{24b}$$

where (24a) is used to get $L_{j+1,j}^\Psi$ by forward substitution whereas $L_{j,j}^\Psi$ is obtained by Cholesky factorization of the left hand side of (24b). The structure-exploiting factorization procedure is summarized in Algorithm 3. The complexity of Alg. 3 is of order $\mathcal{O}(Nn_x^3)$, thereby reducing the complexity from being cubic to linear in the horizon N (compared to the complexity of a generic factorization of the Schur complement with complexity of order $\mathcal{O}(N^3 n_x^3)$). Once L^Ψ is constructed, λ is found by solving (17b) by backward and forward substitution routines exploiting the lower bidiagonal block structure. Finding the search direction d, by solving (17c), is computationally inexpensive as the Cholesky factorizations used in the first step (Alg. 2) can be reused. The solution procedure of (17c) is outlined in Algorithm 4. For the solution of the Newton system in (16) the corresponding optimal step size, $\tau^\star$, exists with a closed-form solution [2, Algorithm 2] under the condition that the equality constraints $Mx = b$ are satisfied, i.e., x is feasible w.r.t. the system dynamics. Further, once x is feasible all subsequent iterates will remain feasible (w.r.t. $Mx = b$) due to linearity. Feasibility can easily be obtained by forward simulation of the system dynamics, e.g., using the shifted solution from a previous time step in the MPC. Alternatively, one can take a full Newton step. Next, we will show how the matrix factorizations can be updated exploiting the OCP structure using a series of rank-one modifications.

A. Low Rank Updates of Matrix Factorizations

In the following we discuss the impact of changes in the active set $\mathcal{J}(x)$ between two inner iterations of Alg. 1. We denote the entering and leaving constraints of the active-set between two successive inner iterations as $\mathcal{J}^e = \mathcal{J}(x^i) \setminus \mathcal{J}(x^{i-1})$ and $\mathcal{J}^l = \mathcal{J}(x^{i-1}) \setminus \mathcal{J}(x^i)$, respectively.

1) Entering Constraints: Consider again the stage-wise generalized Hessian in economic form in (18) for an arbitrary stage and let G_i denote the row i-th row of the bounds defined by $G = \begin{bmatrix} C_j & D_j \end{bmatrix}$ for a single constraint entering the active-set (see (1d)). The element of the generalized Hessian (leaving out the dependency on x for compactness) after the constraint entering the active-set can be expressed as

$$H_{k,j,+} = H_{k,j} + G_i^T \left(\Sigma_{y,k,j}\right)_{ii} G_i = H_{k,j} + \bar{e}_j \bar{e}_j^T \tag{25}$$

where $H_{k,j,+}$ is the element of generalized Hessian after a rank one update given by the entering constraint $\bar{e}_j = G_i^T \sqrt{(\Sigma_{y,k,j})_{ii}} \in \mathbb{R}^{n_x+n_u}$. To understand how the rank one update of the generalized Hessian changes the Schur complement matrix we use the Sherman-Woodbury-Morrison formula

$$\left(A + uv^T\right)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u} \tag{26}$$

which yields

$$H_{k,j,+}^{-1} = H_{k,j}^{-1} - \frac{H_{k,j}^{-1}\bar{e}_j\bar{e}_j^T H_{k,j}^{-1}}{1 + \bar{e}_j^T H_{k,j}^{-1}\bar{e}_j}. \tag{27}$$

Using the existing Cholesky factorization of $H_{k,j}$ to solve the auxiliary linear system $H_{k,j}\hat{e}_j = \bar{e}_j$ and recalling the symmetry of $H_{k,j}^{-1}$ we can rewrite (27) as

$$H_{k,j,+}^{-1} = H_{k,j}^{-1} - \tilde{e}_j\tilde{e}_j^T \tag{28}$$

with $\tilde{e}_j = \frac{\hat{e}_j}{\sqrt{1+\bar{e}_j^T\hat{e}_j}}$. Having defined $H_{k,j,+}^{-1}$, let us consider the resulting changes in Ψ, defined in (19), from the entering constraint. For a change in the active-set of the j-th stage, the sub-block of Ψ

$$\Psi_j \coloneqq \begin{bmatrix} \Psi_{jj} & \Psi_{j,j+1} \\ \Psi_{j,j+1}^T & \Psi_{j+1,j+1} \end{bmatrix} \in \mathbf{Sym}_{++}\left(\mathbb{R}^{2n_x}\right) \tag{29}$$

is the only part that changes. From the entries of Ψ defined in (20) combined with (27) we arrive at

$$\Psi_j^+ = M_j^T H_{k,j,+}^{-1} M_j = \Psi_j - M_j^T\tilde{e}\tilde{e}^T M_j = \Psi_j - \psi_j\psi_j^T \tag{30}$$

where $M_j = \begin{bmatrix} -\mathbf{I}_{\mathbf{n_x}} & \mathbf{0}_{n_x\times n_u} \\ A_j & B_j \end{bmatrix}$ and $\psi_j = M_j^T\tilde{e}_j \in \mathbb{R}^{2n_x}$. Finally, we can express a sparse rank one modification of Ψ as $\psi = \begin{bmatrix} 0 \dots \psi_j \dots 0 \end{bmatrix}$ where ψ_j enters as $\psi_{[j\cdot n_x:(j+2)\cdot n_x]} = \psi_j$.

2) Leaving Constraints: Following the same arguments as for an entering constraint, we arrive at similar rank one modifications considering a single constraint leaving the active-set. That is, $H_{k,j,-} = H_{k,j} - \bar{e}_j\bar{e}_j^T$ defines the generalized Hessian after the constraint leaving the active-set where $\bar{e}_j$ is equivalent to the modification for an entering constraint. The resulting update of Ψ is defined by $\psi_j = M_j^T\tilde{o}_j$ with the modification that $\tilde{o}_j$ is given by $\tilde{o}_j = \frac{\hat{e}_j}{\sqrt{1-\bar{e}_j^T\hat{e}_j}}$ where the sign change in the denominator is caused by the Sherman-Woodbury-Morrison formula. We implement the rank one modifications using [9, Algorithm C1] to modify the factors of $H_k(x)$ and Ψ. Specifically for the rank one modification of Ψ, we have implemented a version exploiting the block-banded structure. Due to the block-wise structure the $H_k(x)$ the rank one modification for one active-set change is computationally very inexpensive, i.e., with complexity $\mathcal{O}((n_x + n_u)^2)$ (since the active-set change only modifies the factor of the block $H_{k,j}(x)$ of $H_k(x)$. Our tailored implementation of the rank one modifications of factors of Ψ yields a worst-case, i.e., active-set change at $j = 0$, complexity of $\mathcal{O}(Nn_x^2)$ and thus is computationally cheaper than recomputing the factorization with Alg. 3. The rank one modifications outlined above can be readily used in between ALM iterations when the penalty parameters are updated. In fact, the resulting modifications are similar to those of an entering constraint with $\bar{e}_j = \begin{bmatrix} C_j & D_j \end{bmatrix}_i^T \sqrt{(\Sigma_{y,k+1,j})_{ii} - (\Sigma_{y,k,j})_{ii}} \in \mathbb{R}^{n_x+n_u}$.

V. Numerical Example

We benchmark the proposed solver against the general purpose version of QPALM [3], OSQP [10], and the tailored OCP solver HPIPM [11]. We use the geometric mean to average the computational times. The numerical experiments are performed on a laptop with Intel(R) Core(TM) i7-10710U @1.10 GHz. Our code is implemented in MATLAB and the results reported are obtained with C-code generated by *codegen*. We consider the classical spring-mass benchmark reported in [8] and often used in literature which allows us to easily assess the numerical performance on problems of different dimensions. The problem is formally stated as:

$$\textbf{minimize} \quad \sum_{j=0}^{N-1}\left(x_j^TQx_j + u_j^TRu_j\right) + x_N^TQx_N \tag{31a}$$

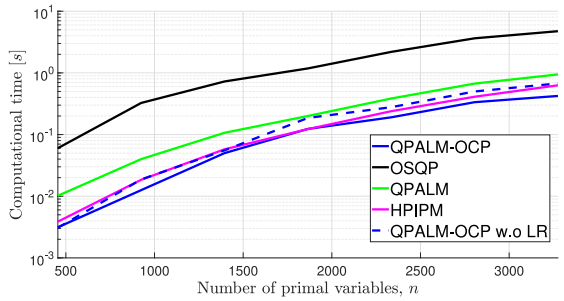
$$\text{s.t.} \quad x_0 = \bar{x}_0 \tag{31b}$$

$$x_{j+1} = Ax_j + Bu_j, \qquad \forall j = 0\cdots N-1 \tag{31c}$$

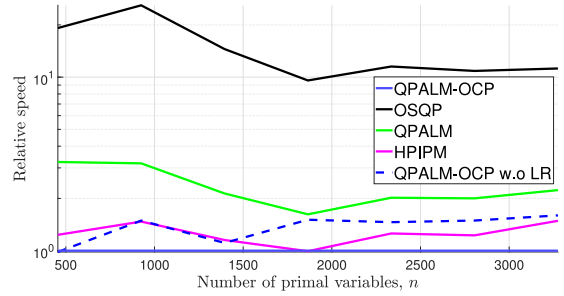
$$-4\cdot\mathbf{1}_{n_x} \le x_j \le 4\cdot\mathbf{1}_{n_x}, \qquad \forall j = 0\cdots N \tag{31d}$$

$$-0.5\cdot\mathbf{1}_{n_u} \le u_j \le 0.5\cdot\mathbf{1}_{n_u}, \quad \forall j = 0\cdots N-1 \tag{31e}$$

where $Q = 10^3\cdot\mathbf{I}_{\mathbf{n_x}}$ and $R_j = 10^{-1}\cdot\mathbf{I}_{\mathbf{n_u}}$. Our solver is aimed at medium to large-scale OCPs, i.e., ranging from hundreds to several thousands of variables. In our numerical experiment we vary the number of masses, M, from 10 to 70 in increments of ten, and thus the number of primal variables ranges from 455 to 3275. For each configuration, we solve 250 OCPs with random initial states. To ensure a wide range of test scenarios, i.e., including both simple problems with few active constraints and more challenging ones with many active constraints, we generate the initial state in the following manner: *(i)* draw $\gamma \sim U[0, 1.0] \in \mathbb{R}$; *(ii)* draw $\bar{x}_0 \sim U\left[-\gamma, \gamma\right] \in \mathbb{R}^{n_x}$. For all solvers, we use the default settings and tolerances equal to 10^{-6}. For a fair comparison, we initialize the primal variables for all solvers with the trajectory obtained by propagating the dynamics with zero input starting from $\bar{x}_0$ (this leads to significant speedups for OSQP, in particular). The optimal solutions found differ on the order of 10^{-5} and constraint violations are comparable. In Fig. 1, the performance is compared with respect to an increasing number of primal variables in terms of computational time. We find that our solver, despite being a non-optimized, prototype implementation, outperforms state-of-the-art general-purpose QP solvers such as OSQP and QPALM. More specifically, we obtain a 13.8 (OSQP), 2.28 (QPALM), and 1.25 (HPIPM)



(a) Computational time with increasing number of primal variables.



(b) Relative speed w.r.t. QPALM-OCP.

Fig. 1. Comparison with QPALM [3], OSQP [10] and HPIPM [11].

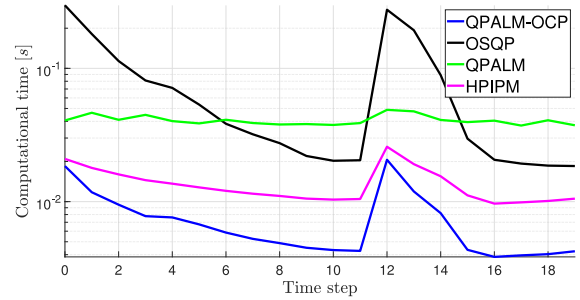


Fig. 2. Closed-loop MPC simulation from 250 random initial states with $M = 20$ and $N = 15$. A large random disturbance is added at simulation step 12.

times speed-up on average, respectively. Further, the benefits from the low rank modifications presented in Section IV-A are illustrated in Fig. 1 by including the results from QPALM-OCP without exploiting low rank modifications. The low rank modifications lead to a 38% speed-up on average. However, in a typical MPC setting, the problem in (31) must be solved repeatedly over time, subject to a new initial condition $\bar{x}_0$. This results in a series of very similar OCPs and therefore the possibility of warm-starting the solver by shifting the previous solution (i.e., primal and dual variables) can be very beneficial. To illustrate this, we run a closed-loop simulation of (31) with $M = 20$ and prediction horizon $N = 15$. Further, we perturb each state with a white Gaussian noise with zero mean and variance 0.01^2 at each simulation step, and at step 12 we add a large random disturbance. We run 250 closed-loop simulations with random initial state generated as described earlier. In Fig. 2, the geometric mean is reported for each time step. Once the MPC law has steered the states close to the origin such that the input constraints are no longer active (see time step nine in Fig. 2), the computational time for all solvers decreases, however, for QPALM-OCP the computational time is significantly lower than the other solvers since it requires very few Newton steps to find the optimal solution and thereby resembling active-set methods in terms of warm-start capabilities. A similar tendency is expected to be seen if the steady state operation has active constraints as the warm-starting would allow finding the new optimum in few iterations even with some changes in the active-set. Similar results to those reported in Fig. 2 are found when testing on problems with different dimensions.

VI. Conclusion and Future Work

In this letter, we presented QPALM-OCP, a structure-exploiting proximal augmented Lagrangian-based QP solver tailored to the constrained optimal control problems arising in Model Predictive Control. A key ingredient is the semismooth Newton method used for subproblem minimization, where structure exploiting factorization updates can be deployed between iterates. This leads to benefits like those of active-set methods, such as cheap iterates and warm-start capabilities. QPALM-OCP is shown to compare favorably against state-of-the-art QP solvers over a wide range of problem dimensions, even though the current implementation is a prototype code-generated from MATLAB and not optimized. In future work, we plan to provide a fully fleshed and efficient C-code implementation and perform more comprehensive benchmarks. Further, QPALM-OCP could be modified to handle other types of multi-stage optimization problems such as Moving Horizon Estimation.

References

[1] R. T. Rockafellar, "Augmented Lagrangians and applications of the proximal point algorithm in convex programming," *Math. Oper. Res.*, vol. 1, no. 2, pp. 97–116, 1976.

[2] B. Hermans, A. Themelis, and P. Patrinos, "QPALM: A Newton-type proximal augmented Lagrangian method for quadratic programs," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, 2019, pp. 4325–4330.

[3] B. Hermans, A. Themelis, and P. Patrinos, "QPALM: A proximal augmented Lagrangian method for nonconvex quadratic programs," *Math. Program. Comput.*, vol. 14, pp. 497–541, Sep. 2022.

[4] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[5] F. Facchinei and J.-S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*. New York, NY, USA: Springer, 2007.

[6] A. Themelis, M. Ahookhosh, and P. Patrinos, *On the Acceleration of Forward-Backward Splitting via an Inexact Newton Method*. Cham, Switzerland: Springer, 2019, pp. 363–412.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.

[8] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[9] P. Gill, G. Golub, W. Murray, and M. Saunders, "Methods for modifying matrix factorizations," *Math. Comp.*, vol. 28, no. 126, pp. 505–535, 1974.

[10] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Math. Program. Comput.*, vol. 12, no. 4, pp. 637–672, 2020.

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

Open Access funding provided by 'Politecnico di Milano' within the CRUI CARE Agreement