Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Accelerated Model Predictive Control Using Restricted Quadratic Programming ⋆

Anson Maitland *,1 **John McPhee** *

* *Department of Systems Design Engineering, University of Waterloo, Waterloo, Canada*

Abstract: We present a method to reduce the computational burden of solving a sequence of convex quadratic programs (QPs). By determining offline what search space is most important, we can restrict our online problem to that subspace, reducing the dimension and computational cost of the QP solver. The process we present is very simple requiring surprisingly little data. Further, we present a modified sequential QP algorithm that leverages the restricted QP approach to solve nonlinear programming problems found in model predictive control. Lastly, we apply these to a benchmark MPC problem and demonstrate their effectiveness using a variety of established QP solvers. We demonstrate that QP problems can be solved faster with minimal MPC performance degradation and highlight future directions for this work.

Keywords: Model-based control, Predictive control, Quadratic programming, Nonlinear programming

1. INTRODUCTION

Model Predictive Control (MPC) is a modern feedback control strategy that requires the online solution of a parameterized optimization problem within a fixed sampling time (Rawlings and Mayne, 2009). In MPC, one predicts the plant's state over a finite horizon of H steps using the latest measurement (or estimate) of the plant's state and optimizes the future controller actions. Only the first control action is implemented and the process is repeated at succeeding timesteps. The resulting sequence of optimization problems (and their solutions) are thus intimately connected by the evolution of the plant and controller action history.

MPC has garnered significant attention in the academic and industrial control communities as it can handle a wide variety of problems (Garcia et al., 1989; Qin and Badgwell, 1997, 2003; Lee, 2011). Although MPC allows one to explicitly handle nonlinear dynamics, constraints and other problem aspects not possible in classical control, the hard real-time constraint of many real systems, which require fast sampling rates, presents a significant challenge to MPC implementation. In response, significant attention has been devoted to algorithms dedicated to the rapid solution of optimal control problems found in MPC, *e.g.* FORCES NLP: a fast interior-point method (Domahidi et al., 2012; Zanelli et al., 2017), CVXGEN: a convex optimizer that can auto-generate custom C code for embedded applications (Mattingley et al., 2010; Mattingley and Boyd, 2012) and the real-time iteration scheme which is an MPC approximation method (Diehl et al., 2002, 2005). For linear systems, a variety of fast convex solvers have been developed that run at the millisecond timescale for MPC applications (Richter et al., 2011; Domahidi et al., 2012; Jerez et al., 2014; Frison et al., 2014; Ferreau et al., 2014).

The solution of convex quadratic programs (QPs) has been given special attention as these problems form the backbone of many optimization algorithms for nonconvex, nonlinear and mixed-integer problems. Broadly, there are three distinct classes of QP solvers: active set, interior point and first order methods. Respective examples are: qpOASES which uses a homotopy based strategy (Ferreau et al., 2014), OOQP which has an object-oriented design (Gertz and Wright, 2003) and OSQP which uses the Alternating Direction Method of Multipliers (ADMM) algorithm (Stellato et al., 2017). This paper introduces a QP reduction strategy, called restricted quadratic programming (rQP), to reduce QP solution times that is applicable to all three classes of solver.

The underlying idea behind the rQP approach was first presented by the authors in the context of Newton's method as applied to the Karush-Kuhn-Tucker conditions of an unconstrained optimization problem (Maitland and McPhee, 2017; Maitland et al., 2019). By wisely selecting a subspace of the total search space, one can reduce the computational burden of finding a solution without degrading the solution quality. This approach is particularly well-suited for scenarios where one has to solve a sequence of similar optimization problems, such as MPC. The subspace selection is done in an offline stage using simulation data from a collection of representative test cases. In practice, the amount of data required has proven to be surprisingly small and the selection strategy boils down to simple matrix algebra.

This paper is organized in the following manner: firstly, in Section 2 we review the basics of QP, introduce rQP and

⋆ This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Ontario Centres of Excellence (OCE) and Maplesoft™.

1 corresponding author, email: anson.maitland@uwaterloo.ca

Copyright lies with the authors

illustrate how one can select a good subspace by analyzing the error introduced via rQP. In Section 3, we demonstrate how one can integrate rQP into a general nonlinear programming (NLP) method by modifying a standard sequential QP (SQP) algorithm. Lastly, in Section 4 we apply rQP to a benchmark MPC problem followed by some concluding remarks in Section 5.

Notation: matrices are denoted by capital letters, *e.g.* $A, B, \ldots$ and vectors by lowercase letters, *e.g.* $x, y, \ldots$ All vectors are considered column vectors. We denote the ith row of a matrix or element of a vector by A_i or x_i, respectively. Further, $A_{\mathcal{I}}$ denotes the matrix made up of the rows of A belonging to the index set $\mathcal{I}$. We denote the span of the columns of matrix A by $\langle A \rangle$. Unless specified otherwise all norms are 2-norms or induced 2-norms.

2. RESTRICTED QUADRATIC PROGRAMMING

In this section we begin with a review of convex QPs. These are a backbone to many NLP methods and are found frequently in the MPC setting for linear and nonlinear systems. We then introduce *restricted* QPs and, following an error analysis, introduce a simple restricted subspace learning method.

2.1 Quadratic Programming Review

In general, a strictly convex QP can be written as

$$\begin{aligned} \min_{y} \quad & \frac{1}{2} y^T G y + g^T y \\ \text{subject to} \quad & Ay + b \geq 0 \end{aligned} \tag{1}$$

where G is positive definite. For purposes of discussion we assume the feasible region is nonempty so Problem (1) has a unique solution that we denote y^*. This solution can be recovered from the solution of the following (possibly smaller) unconstrained QP

$$\min_{z} \quad \frac{1}{2} z^T G' z + g'^T z \tag{2}$$

where $G' = N^T G N$ and $g' = N^T(g + GRp)$. N is the kernel of $A_{\mathcal{A}(y^*)}$ where $\mathcal{A}(y^*) = \{i | A_i y^* + b_i = 0\}$ is the set of active constraints at y^*, such that $A_{\mathcal{A}(y^*)} N = 0$. The solutions are related by the relation,

$$y^* = N z^* + Rp$$

where R is chosen such that the matrix $[N \ R]$ is nonsingular and p is any solution to $A_{\mathcal{A}(y^*)} Rp + b_{\mathcal{A}(y^*)} = 0$. The solution of Problem (2) can be written as $z^* = -G'^{-1} g'$. We call Problem (2) the *unconstrained* transformation of Problem (1).

In the transformation of Problem (1) we restrict our attention to active constraints, yielding linear equality constraints that we then eliminate (Wright and Nocedal, 1999; Boyd and Vandenberghe, 2004). Problem (2) can be understood as the Null-Space Method applied to Problem (1) about the optimum y^*. It is important to note that the Problem (2) was constructed knowing y^*, the solution of Problem (1). In practice, this means that we cannot solve our original problem using the equivalent unconstrained version directly. For example, in active-set methods one 'learns' $\mathcal{A}(y^*)$ during the iterative solution process. We will return to this incongruity in subsequent sections.

2.2 Introducing Restrictions

Suppose we knew the solution y^* of Problem (1) then not only could we write down an explicit expression for the solution, knowing the active constraints $\mathcal{A}(y^*)$, but we could stretch this idea further and introduce new constraints such that the solution remains the same. All that will change is the manner in which we can explicitly express y^*. We shall see how this is useful in the following.

A *restricted* version of QP (1) is any QP of the form

$$\begin{aligned} \min_{y} \quad & \frac{1}{2} y^T G y + g^T y \\ \text{subject to} \quad & Ay + b \geq 0 \\ & \Pi_{|r}^{\perp T} y = 0 \end{aligned} \tag{3}$$

where $\Pi_{|r}^{\perp}$ is a matrix specifying the introduced linear equality constraints. As long as $y^* \in \langle \Pi_{|r} \rangle$, where $\Pi_{|r}$ is the kernel of $\Pi_{|r}^{\perp}$, then the solution of (1) and (3) are the same. Without loss of generality we can assume that $\Pi = \begin{bmatrix} \Pi_{|r} & \Pi_{|r}^{\perp} \end{bmatrix}$ is an orthogonal matrix, and $\Pi_{|r}$ simply denotes the first r columns of Π. To be of interest we assume the feasible set when the constraints are added is nonempty.

Eliminating the constraints of Problem (3) yields the rQP

$$\begin{aligned} \min_{z} \quad & \frac{1}{2} z^T [\Pi_{|r}^T G \Pi_{|r}] z + g^T \Pi_{|r} z \\ \text{subject to} \quad & A \Pi_{|r} z + b \geq 0 \end{aligned} \tag{4}$$

which is a QP of order r. The solution can be expressed as

$$z_{|r}^* = -\Pi_{|r} [\Pi_{|r}^T G' \Pi_{|r}]^{-1} \Pi_{|r}^T g' \tag{5}$$

if $\mathcal{A}(y^*) = \mathcal{A}(y_{|r}^*)$. We note that if we consider general $\Pi_{|r}$ the solution of (4) may not have the same active constraints at its optimum as the unrestricted problem (1) and the expression given by equation (5) will be incorrect. It is possible that the solutions differ by an arbitrarily large amount due to the presence of inequality constraints. This occurs when constraints near y^* are nearly collinear with the span of $\Pi_{|r}$. However, by choosing $\Pi_{|r}$ well one can avoid these pathological situations.

Now in practice if one is solving a QP they would not know y^* and hence an appropriate choice of $\Pi_{|r}$. If we suppose $\mathcal{A}(y^*) = \mathcal{A}(y_{|r}^*)$ then we can easily examine the error between z^* and $z_{|r}^*$. We can do the analysis by breaking the error into two orthogonal components using the projection operator $P_{|r} = \Pi_{|r} \Pi_{|r}^T$ and its orthogonal complement $P_{|r}^{\perp} = \Pi_{|r}^{\perp} \Pi_{|r}^{\perp T} = I - P_{|r}$. The error between the true solution and the approximate one is

$$\begin{aligned} \|z^* - z_{|r}^*\|^2 &= \|P_{|r}(z^* - z_{|r}^*)\|^2 + \|P_{|r}^{\perp}(z^* - z_{|r}^*)\|^2 \\ &= \|\Pi_{|r}([\Pi_{|r}^T G' \Pi_{|r}]^{-1} \Pi_{|r}^T - \Pi_{|r}^T G'^{-1}) g'\|^2 \\ &\quad + \|P_{|r}^{\perp} z^*\|^2 \\ &= \|\Pi_{|r} [\Pi_{|r}^T G' \Pi_{|r}]^{-1} \Pi_{|r}^T (I - G' P_{|r} G'^{-1}) g'\|^2 \\ &\quad + \|P_{|r}^{\perp} z^*\|^2 \\ &\leq (\|\Pi_{|r} [\Pi_{|r}^T G' \Pi_{|r}]^{-1} \Pi_{|r}^T G'\|^2 + 1) \|P_{|r}^{\perp} z^*\|^2 \\ &\leq (\kappa(G')^2 + 1) \|P_{|r}^{\perp} z^*\|^2 \end{aligned}$$

where $\kappa(\cdot)$ is the condition number of of its matrix argument. Thus if we can choose $\Pi_{|r}$ such that $\|P_{|r}^{\perp} z^*\|^2$ is small, then we can guarantee that the solution $z_{|r}^*$ is a good approximation. In the following section we present a strategy to do just this.

2.3 Subspace Learning

In practice, we are often not interested in solving a single QP but a sequence of related QPs to solve some more general optimization problem, *e.g.* SQP. Furthermore, one often chooses the solution of the preceding QP as the initial guess of the current QP. In this case we can view the solution of the kth QP $z^*[k]$ as

$$z^*[k] = \sum_{j=1}^{k} \Delta z[j] + z^*[0]$$

where $z^*[0]$ is the initial guess of the very first problem which without loss of generality we can assume to be 0 and $\Delta z[j] = z^*[j] - z^*[j-1]$. Then we can bound the error by

$$\|P_{|r}^{\perp} z^*[k]\|^2 \leq \sum_{j=1}^{k} \|P_{|r}^{\perp} \Delta z[j]\|^2.$$

To minimize this bound it is optimal to choose $\Pi_{|r}$ as the solution of

$$\begin{aligned} \min_{A \in \mathbb{R}^{n \times r}} \quad & \sum_{j=1}^{k} \|(I - AA^T)\Delta z[j]\|^2 \\ \text{subject to } & A^T A = I. \end{aligned} \tag{6}$$

The solution A^* of (6) can be expressed straightforwardly using the singular value decomposition (SVD) of the matrix $S = [\Delta z[1] \cdots \Delta z[k]]$. If we denote the SVD by $S = \Pi \Sigma V^T$ then A^* is exactly the first r columns of Π.

Again, in practice one will not know any of the solutions of the QP sequence ahead of time (else they would not need to be computed) so we cannot select $\Pi_{|r}$ using the above. But in scenarios where we can compute the solutions of representative sequences of QPs offline, such as MPC simulations, we can use that information to find a good approximation to the optimal choice of $\Pi_{|r}$ and then use that choice online to speed up QP computation times using rQP.

We can compute a good choice of $\Pi_{|r}$ using the following steps:

(1) Run a set of representative QP sequences to collect the snapshot matrix S by concatenating the results.
(2) Compute the SVD $S = \Pi \Sigma V^T$.
(3) Select $\Pi_{|r}$ from the first r columns of Π.

Ideally, one wants to choose r as small as possible in order to accelerate the QP computation time as much as possible. However, depending on the problem not all values of r may lead to good solutions. Analytically determining a good choice for r is an open problem and is currently done by trial and error to find an acceptable balance between computation time and approximate solution quality.

3. A RESTRICTED SQP METHOD FOR MPC

In this section we present a modified SQP method, we call restricted SQP (rSQP), that takes advantage of rQP shown in Section 2. The novelty of this method is to combine the solutions of rQP subproblems with orthogonal gradient descent steps.

Consider a generic MPC problem

$$\begin{aligned} \min_{\underline{u}} \quad & J(\underline{u}; x) \\ \text{subject to} \quad & C(\underline{u}; x) \geq 0 \end{aligned}$$

where J is the objective (or cost) function, $\underline{u}$ is the vector of controls over the horizon, x is the plant's state and C are constraints. We have presented the problem compactly by assuming only the controls are optimization variables thus simplifying the presentation. This could be accomplished by eliminating the dynamic constraints using a single-shooting formulation. In future work, we shall show how both primal and dual variables can be handled by the rQP approach.

The rSQP method solves a sequence of QP subproblems using rQP in combination with a line search strategy based on merit functions. The QP subproblems are

$$\begin{aligned} \min_{\Delta \underline{u}} \quad & \frac{1}{2} \Delta \underline{u}^T G \Delta \underline{u} + g^T \Delta \underline{u} \\ \text{subject to} \quad & A \Delta \underline{u} + b \geq 0 \end{aligned} \tag{7}$$

where $\Delta \underline{u}$ is the variable update step so that $\underline{u}^+ = \underline{u} + \Delta \underline{u}$, $G = \nabla^2 J(\underline{u}; x)$, $g = \nabla J(\underline{u}; x)$, $A = \nabla C(\underline{u}; x)$ and $b = C(\underline{u}; x)$. To solve Problem (7) we break down the solution into two orthogonal components $\Delta \underline{u}_-$ and $\Delta \underline{u}_\perp$. The first component is found by solving the rQP given in (4) where the matrix $\Pi_{|r}$ was found using the method of Section 2.

Though we have determined that $\Pi_{|r}$ is likely the most 'important' subspace, we must take orthogonal steps to guarantee global convergence and ultimately garner good performance (Maitland et al., 2019). These orthogonal steps are computed cheaply using a projection of the gradient

$$\Delta \underline{u}_\perp = -P_{|r}^{\perp} g^T.$$

This step is scaled by a factor γ that is determined by optimizing the step length. The full update of one iteration is

$$\Delta \underline{u} = \alpha(\Delta \underline{u}_- + \gamma \Delta \underline{u}_\perp)$$

where α is given by a line search. Plugging this expression back into the cost function of Problem (7) we get a quadratic optimization problem in γ

$$\min_{\gamma} \quad \frac{\gamma^2}{2} \Delta \underline{u}_\perp^T G \Delta \underline{u}_\perp + \gamma(\Delta \underline{u}_\perp^T G \Delta \underline{u}_- + \frac{1}{\alpha} g^T \Delta \underline{u}_\perp)$$

which has an explicit solution. Since the line search is actually conducted after the determination of the two orthogonal components (this is necessary for global convergence guarantees (Wright and Nocedal, 1999; Boyd and Vandenberghe, 2004)) α is unknown. In our implementation we optimistically set $\alpha = 1$ to find γ.

The final step is the line search used to guarantee convergence and not violate constraints. The line search is a standard backtracking algorithm that uses the ℓ_1 merit function

$$\phi(\underline{u}; x, \mu) = J(\underline{u}; x) + \frac{1}{\mu}\|C(\underline{u}; x)|_{\mathcal{A}}\|_1$$

with directional derivative

$$d(\phi(\underline{u}; x, \mu); \Delta\underline{u}) = \nabla J(\underline{u}; x)\Delta\underline{u} + \frac{1}{\mu}\|C(\underline{u}; x)|_{\mathcal{A}}\|_1.$$

Algorithm 1 Restricted SQP

1: **procedure** RSQP($\underline{u}, x, \mathcal{A}; \Pi_{|r}$)
2: **repeat**
3: $G \leftarrow \nabla^2 J(\underline{u}; x)$
4: $g \leftarrow \nabla J(\underline{u}; x)$
5: $A \leftarrow \nabla C(\underline{u}; x)$
6: $b \leftarrow C(\underline{u}; x)$
7: $(z^*_{|r}, \mathcal{A}, \lambda) \leftarrow \mathrm{QP}(\Pi^T_{|r} G \Pi_{|r}, g\Pi_{|r}, A\Pi_{|r}, b, \mathcal{A})$
8: $\Delta\underline{u}_- \leftarrow \Pi_{|r} z^*_{|r}$
9: $\Delta\underline{u}_\perp \leftarrow -(\mathbf{I} - \Pi_{|r}\Pi^T_{|r})g^T$
10: $\gamma \leftarrow \frac{-(\Delta\underline{u}_\perp^T G \Delta\underline{u}_- + g^T \Delta\underline{u}_\perp)}{\Delta\underline{u}_\perp^T G \Delta\underline{u}_\perp}$
11: $\alpha \leftarrow$ LineSearch($\underline{u}, \Delta\underline{u}_- + \gamma\Delta\underline{u}_\perp, \mathcal{A}, \|\lambda\|_\infty$)
12: $\Delta\underline{u} \leftarrow \alpha(\Delta\underline{u}_- + \gamma\Delta\underline{u}_\perp)$
13: $\underline{u} \leftarrow \underline{u} + \Delta\underline{u}$
14: **until** Termination
15: **return** $\underline{u}$
16: **end procedure**

1: **procedure** LineSearch($\underline{u}, \Delta\underline{u}, \mathcal{A}, \mu$)
2: $\alpha \leftarrow 1$
3: **while** $\phi(\underline{u} + \alpha\Delta\underline{u}; x, \mu + \delta') > \phi(\underline{u}; x, \mu + \delta') + \eta\alpha d(\phi(\underline{u}; x, \mu + \delta'); \Delta\underline{u})$ **do**
4: $\alpha \leftarrow \tau\alpha$
5: **end while**
6: **return** α
7: **end procedure**

The full rSQP method is presented in Algorithm 1. The novelty of this algorithm is in lines $7 - 11$ which break down the QP solution into orthogonal components. The remaining portions of the algorithm are based on Chapter 18 of (Wright and Nocedal, 1999). The routine QP represents a generic QP solver. One will note that the matrix input $\Pi^T_{|r} G \Pi_{|r}$ may no longer be sparse but will retain the definiteness of G since $\Pi_{|r}$ is a subset of the columns of an orthogonal matrix. The loss of sparsity will have an impact on the computational burden of most QP solvers. In the line search method η, τ, δ' are small positive constants selected by the user. We selected $\eta = 0.1, \tau = 0.5, \delta' = 10^{-3}$ in our simulations. We also selected $J(\underline{u} - \Delta\underline{u}; x) - J(\underline{u}; x) < 10^{-6}$ as our termination criteria.

4. CASE STUDY

In this section we demonstrate the rSQP using an MPC benchmark problem. The plant is an unstable system given by

$$\dot{x_1} = x_2 + u(\mu + (1 - \mu)x_1)$$
$$\dot{x_2} = x_1 + u(\mu - 4(1 - \mu)x_2)$$

with $\mu = \frac{1}{2}$. Our MPC is formulated using single-shooting with a forward Euler discretization of the plant. The goal of the MPC is to drive the plant to zero while minimizing

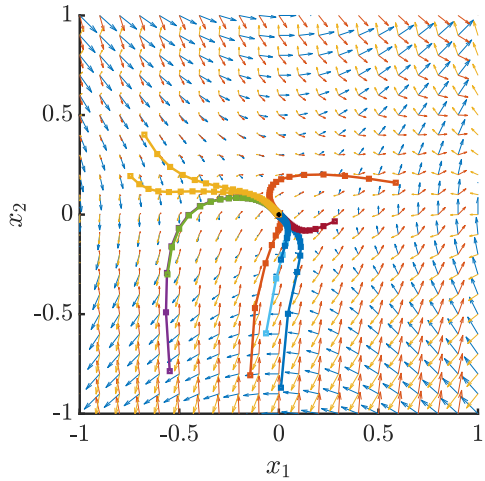

Fig. 1. The state trajectories of ten MPC simulations used to generate the snapshot data. The vector field of the plant is shown in the background (blue $u = 0$, red $u = 2$, yellow $u = -2$).

the control effort. The MPC's parameterized cost is given by

$$J(\underline{u}; x(0)) = \sum_{k=0}^{H-1} \|x(k)\|^2_Q + \|u(k)\|^2_R + \|x(H)\|^2_P$$

where $\underline{u} = [u(0), u(1), \ldots, u(H-1)]$ is the control over the horizon, $x(0)$ is the latest measurement of the plant's state and $x(k)$ are the predicted states. The only constraints on the problem are control bounds $u \in [-2, 2]$ and a terminal constraint $\|x(H)\|^2_P \leq \alpha$. The prediction and the control horizon are equal at $H = 15$ timesteps. This problem is presented in (Quirynen et al., 2015) and is further discussed in (Chen and Allgöwer, 1998). For the full list of problem parameters used, see Case A in Section 5 of (Quirynen et al., 2015).

We selected 10 random initial conditions within $[-1, 1]^2$ to generate our snapshot data; these are shown in Figure 1. We then applied our SVD based method to determine $\Pi_{|r}$ for $1 \leq r \leq 15$. In Figure 2 we plot the state trajectories of five example MPC simulations using rSQP with $r = 2$. From Figure 2 we can see that the quality of the MPC solution using rSQP solution is sensitive to the initial conditions. To better measure the quality of the rSQP solutions we plot the cost of the entire simulation $J_{|r} = \sum \|x(k)\|^2_Q + \|u(k)\|^2_R$, where $x(k)$ is the state trajectory and $u(k)$ the implemented controller actions, for all values of r in Figure 3. Firstly, we note that the quality of solution is not always monotonically increasing with r contrary to what one might expect. Some solutions have consistently good solutions while others unexpectedly degrade for particular values of r. However, despite the initial condition sensitivity all sample trajectories had solutions within 1% of the unrestricted cost J^* for some small values of r. These results bring up interesting questions regarding the robustness of the rSQP method and its interaction with constraints.

Further, to better understand the rSQP method as a solver, one must look at its convergence properties. In

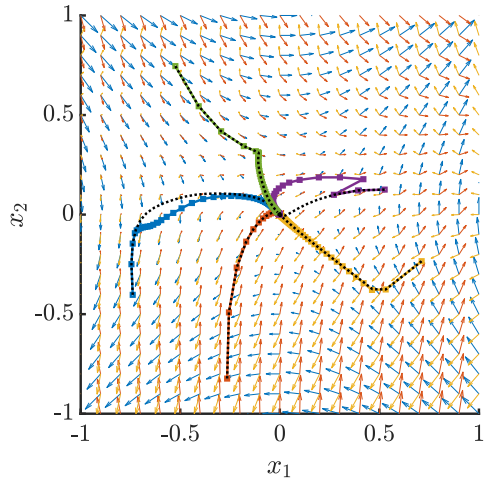

Fig. 2. MPC controlled state trajectories using the rSQP solver with $r = 2$. The state trajectories using an unrestricted solver are shown in dotted black.

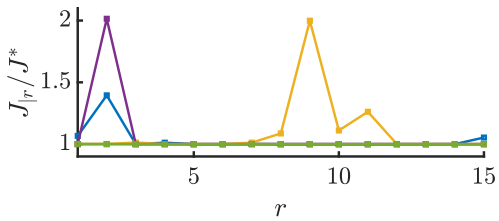

Fig. 3. The normalized cost of the five selected trajectories shown in Figure 2 for different orders of restriction.

Figure 4 we display the convergence behaviour of rSQP for different values of r on a single problem. From this plot

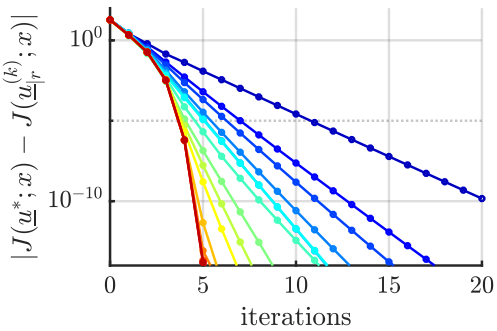

Fig. 4. Convergence behaviour of rSQP for a sample problem with $x = [-0.2, 0.2]^T$, $\underline{u}^{(0)} = [1, 1, \ldots, 1]^T$. The true solution is $\underline{u}^*$ and $\underline{u}^{(k)}_{|r}$ is the rSQP's k^{th} iteration. The coloured lines correspond to different values of r ranging from $r = 1$ (dark blue) to $r = 15$ (dark red).

we can see that as r increases we recover the quadratic convergence behaviour of Newton's method, as expected. However for smaller values of r we can still acquire superlinear convergence and acquire highly accurate solutions in only a few iterations.

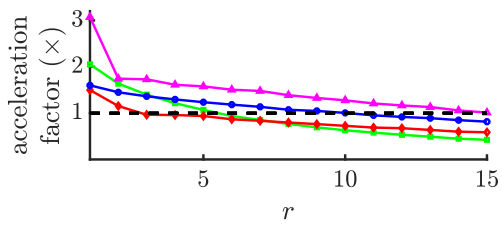

Fig. 5. Mean acceleration factor of QP solutions for 100 simulations. OOQP: red diamond, OSQP: green square, qpOASES: blue circle, ASQP: magenta triangle.

Lastly, we measured the computational acceleration garnered by the rQP approach with 4 different QP solvers. The first is an active set method written by the authors inspired by Chapter 16 of (Wright and Nocedal, 1999) which we call ASQP, the second is qpOASES another active set method that is particularly well-suited for MPC problems (Ferreau et al., 2014, 2008), third is OSQP a convex QP solver that was only recently released with promising results for MPC applications (Stellato et al., 2017), and lastly OOQP an interior point method based QP solver (Gertz and Wright, 2003).

Both qpOASES and OSQP were run using the MATLAB interface and OOQP was run using the OPTI toolbox for MATLAB (Currie et al., 2012). They were run utilizing their problem setup and update features for faster execution. We did not modify the default solver parameters. All simulations were run using MATLAB R2019a on a desktop computer with an Intel Core i7-4790 CPU with Realtime priority. The derivatives of the objective function and constraints were computed symbolically using Maple and exported in optimized form for use with MATLAB.

We compared the computation times of the rQP solutions to an unrestricted QP. From Figure 5 we can observe that OOQP, OSQP and qpOASES take advantage of the sparsity of the unrestricted QP and can solve the larger unreduced problem faster than some of the restricted QPs for larger values of r. The black dashed line is the mean QP solution time without any restriction for 100 simulations (for reference these values are: OOQP 285.66μs, OSQP 40.405μs, qpOASES 15.745μs, ASQP 8.0616μs). However, for all QP solvers tested, the reduced QP approach can yield computational speedups for small enough r. The greatest acceleration comes from solvers that don't exploit sparsity since the restricted QP elements become non-sparse. We note that the acceleration factor can increase even further for larger problems.

5. CONCLUSION

To reduce the computational burden of QPs we have introduced a restricted QP approach. From our case study we have seen that this approach delivers greatest computational benefit for those QP algorithms that do not rely heavily on structured sparse matrix operations. As all strategies can typically benefit from exploiting sparsity, future work should consider modifying the subspace learning process to produce sparse restriction matrices. One possible direction would be to to use the snapshot data to produce near optimal move-blocking strategies

for nonlinear systems (Shekhar and Manzie, 2015). Other options to improve computational gains would be to use a restricted version of a quasi-Newton method to update the Hessian and cheaper γ selection strategies.

We demonstrated the benefits of rQP for an MPC problem formulated using single-shooting and thus restricted only the space of primal variables. In future work, we will demonstrate how we can expand the subspace learning approach to restrict both primal and dual variables for other formulations of MPC problems. This would extend the computational benefits that can be achieved using our method to other types of solvers.

The success of rQP as an approximate QP solver is entirely dependent on the choice of restricting subspace. We have introduced a method to select this subspace using an offline processing stage. However, it remains an open question as to how to optimally choose snapshot data to garner good, robust MPC performance using the SVD-based method we present. Future questions ripe for investigation include: what are the key ingredients one needs in their snapshot data to guarantee MPC robustness? how sensitive is the subspace selection to the snapshot data? and ultimately how sensitive is the MPC performance to the choice of subspace?

REFERENCES

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge university press.

Chen, H. and Allgöwer, F. (1998). A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10), 1205–1217.

Currie, J., Wilson, D.I., et al. (2012). OPTI: lowering the barrier between open source optimizers and the industrial MATLAB user. *Foundations of computer-aided process operations*, 24, 32.

Diehl, M., Bock, H.G., Schlöder, J.P., Findeisen, R., Nagy, Z., and Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4), 577–585.

Diehl, M., Bock, H.G., and Schlöder, J.P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5), 1714–1736.

Domahidi, A., Zgraggen, A.U., Zeilinger, M.N., Morari, M., and Jones, C.N. (2012). Efficient interior point methods for multistage problems arising in receding horizon control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, 668–674. IEEE.

Ferreau, H.J., Bock, H.G., and Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 18(8), 816–830.

Ferreau, H.J., Kirches, C., Potschka, A., Bock, H.G., and Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4), 327–363.

Frison, G., Sørensen, H.B., Dammann, B., and Jørgensen, J.B. (2014). High-performance small-scale solvers for linear model predictive control. In *Control Conference (ECC), 2014 European*, 128–133. IEEE.

Garcia, C.E., Prett, D.M., and Morari, M. (1989). Model predictive control: theory and practice – a survey. *Automatica*, 25(3), 335–348.

Gertz, E.M. and Wright, S.J. (2003). Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software (TOMS)*, 29(1), 58–81.

Jerez, J.L., Goulart, P.J., Richter, S., Constantinides, G.A., Kerrigan, E.C., and Morari, M. (2014). Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12), 3238–3251.

Lee, J.H. (2011). Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, 9(3), 415.

Maitland, A., Jin, C., and McPhee, J. (2019). The restricted Newton method for fast nonlinear model predictive control. In *ASME 2019 Dynamic Systems and Control Conference*, in print. American Society of Mechanical Engineers.

Maitland, A. and McPhee, J. (2017). Improving model predictive controller turnaround time using restricted Lagrangians. In *Decision and Control (CDC), 2017 IEEE 56th Annual Conference on*, 3811–3816. IEEE.

Mattingley, J. and Boyd, S. (2012). Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1), 1–27.

Mattingley, J., Wang, Y., and Boyd, S. (2010). Code generation for receding horizon control. In *2010 IEEE International Symposium on Computer-Aided Control System Design*, 985–992. IEEE.

Qin, S.J. and Badgwell, T.A. (1997). An overview of industrial model predictive control technology. In *AIche Symposium Series*, volume 93, 232–256. New York, NY: American Institute of Chemical Engineers, 1971-c2002.

Qin, S.J. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control engineering practice*, 11(7), 733–764.

Quirynen, R., Vukov, M., and Diehl, M. (2015). Multiple shooting in a microsecond. In *Multiple Shooting and Time Domain Decomposition Methods*, 183–201. Springer.

Rawlings, J.B. and Mayne, D.Q. (2009). *Model predictive control: Theory and design*. Nob Hill Pub.

Richter, S., Jones, C.N., and Morari, M. (2011). Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. *IEEE Transactions on Automatic Control*, 57(6), 1391–1403.

Shekhar, R.C. and Manzie, C. (2015). Optimal move blocking strategies for model predictive control. *Automatica*, 61, 27–34.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2017). OSQP: An operator splitting solver for quadratic programs. *arXiv e-prints*.

Wright, S.J. and Nocedal, J. (1999). *Numerical Optimization*, volume 2. Springer New York.

Zanelli, A., Domahidi, A., Jerez, J., and Morari, M. (2017). FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, 1–17.