A STABLE AND EFFICIENT METHOD FOR SOLVING A CONVEX QP WITH APPLICATION TO OPTIMAL CONTROL*

AMIR SHAHZAD†, ERIC C. KERRIGAN‡, AND GEORGE A. CONSTANTINIDES§

Abstract. A method is proposed for reducing the cost of computing search directions in an interior point method for a quadratic program. The KKT system is partitioned and modified, based on the ratios of the slack variables and dual variables associated with the inequality constraints, to produce a smaller, approximate linear system. Analytical and numerical results are included that suggest the distribution of eigenvalues of the new, approximate system matrix is improved, which makes it more amenable to being solved with an iterative linear solver. For this purpose, new preconditioners are also presented to allow iterative methods, such as MINRES, to be used. Numerical results indicate that the computational complexity of the proposed method scales well when applied to a finite horizon discrete-time optimal control problem with linear dynamics, quadratic cost and linear inequality constraints, which arises in model predictive control applications.

Key words. Interior point methods, Quadratic programming, Ill-conditioning, Large-scale problems, Inexact methods, Iterative methods, Predictive control

AMS subject classifications. 90C51, 90C20, 90C06, 15A12, 49M15, 49N05

1. Introduction.

Interior-point methods have proven to be an efficient way of solving linear, quadratic and nonlinear programming problems. Quadratic programs (QPs) arise in many applications, such as least-squares regression with linear constraints, robust data fitting, support vector machines and predictive control problems [5,14,15,21]. They also appear in solving a nonlinear programming problem with sequential quadratic programming, in which a series of QPs is solved [24, Chap. 18].

Many primal-dual interior point methods (IPMs) for solving a QP involve finding a solution to the Karush-Kuhn-Tucker (KKT) conditions by a Newton-type method [30], in which a linear system of equations is formed and solved at each IPM iteration. Solving this linear system is the main contributor to the total computational cost of an IPM.

The linear system that arises in this process is often sparse and large. Its size is usually reduced by block elimination, which provides alternative linear systems, as reviewed in Section 2. However, the reduction in size may adversely affect the sparsity of the system matrix. In Section 3 we present a new method that replaces the linear system with a smaller, approximate linear system, in which sparsity has been preserved. An upper bound on the error introduced, which decreases with each IPM iteration, is also presented. A nice property of this approximate linear system is that it can be reduced, if desired, to a linear system whose size converges to the number of active constraints at the solution.

Though the proposed linear system can be solved with direct methods, we particularly investigate the use of iterative methods, such as the minimum residual (MINRES) method [25], for three reasons [8]: First, iterative methods have a higher ratio

*This work was funded by the EPSRC under grant number EP/G031576/1 and the European Union Seventh Framework Programme under grant agreement number FP7-ICT-2009-4 248940 (EMBOCON)

†Department of Engineering Management, Centre for Advanced Studies in Engineering, 19-Attaturk Avenue, G-5/1, Islamabad, Pakistan. (a.shahzad199@gmail.com).

‡Department of Aeronautics and Department of Electrical and Electronic Engineering, Imperial College London, Exhibition Road, SW7 2AZ, U.K. (e.kerrigan@imperial.ac.uk)

§Department of Electrical and Electronic Engineering, Imperial College London, Exhibition Road, SW7 2AZ, U.K. (g.constantinides@imperial.ac.uk)

of addition and multiplication operations to division and square root operations, compared to direct methods, and hence are more efficient from a hardware point of view. Second, iterative methods can more easily exploit sparsity compared to direct methods. Third, in iterative methods one can trade-off accuracy with computational time, whereas this is not possible with direct methods.

Inexact Newton methods have been proposed to reduce the computational effort in the solution of the optimality conditions [10, 11, 13]. The main idea of these methods is to terminate the iterative linear solver early with less accuracy when the initial iterations are far from the optimal point. Therefore, direct methods, which solve the linear system accurately, are not applicable to inexact Newton methods. In Section 3 we investigate the efficacy of iterative methods for solving the newly proposed linear system within an inexact IPM.

An important point to consider with iterative linear solvers is that the linear system to be solved becomes ill-conditioned at later iterations in the IPM. We present some results in Section 3 that suggest that the distribution of eigenvalues of the matrix of the smaller, approximate linear system is not necessarily worse than for the original system. The rate of convergence of the iterative solver can also be enhanced with new preconditioners introduced in Section 4.

Section 5 gives a detailed analysis of the computational complexity of using the smaller, approximate linear system in an IPM, compared to using the original linear system.

Section 6 applies the new method to finite horizon discrete-time optimal control problems with linear dynamics, quadratic cost and linear inequality constraints. Numerical simulations indicate that the computational complexity of the new method compares favorably with existing approaches.

2. Review of exact and inexact interior point methods. Consider a convex QP of the form

$$x^* := \arg\min_x \frac{1}{2}x^T H x + h^T x \tag{2.1a}$$

subject to

$$\underline{x}_i \leq x_i \quad \forall i \in \mathcal{I}_l, \quad x_i \leq \overline{x}_i \quad \forall i \in \mathcal{I}_u \tag{2.1b}$$

$$Dx \leq d, \quad Fx = f, \tag{2.1c}$$

where $x \in \mathbb{R}^{n_d}, h \in \mathbb{R}^{n_d}, d \in \mathbb{R}^{n_i}, f \in \mathbb{R}^{n_e}, H \in \mathbb{R}^{n_d \times n_d}, F \in \mathbb{R}^{n_e \times n_d}, D \in \mathbb{R}^{n_i \times n_d}$, n_d is the number of decision variables, n_e is the number of equality constraints, n_i is the number of inequality constraints that cannot be expressed in terms of simple lower and upper bounds on x, and H is positive semidefinite. Here $\mathcal{I}_l$ and $\mathcal{I}_u$ are index sets and we define these sets as $\mathcal{I}_l := \{p_1, p_2, \ldots, p_{n_l}\}$, $\mathcal{I}_u := \{q_1, q_2, \ldots, q_{n_u}\}$. We define rectangular matrices $P_l \in \mathbb{R}^{n_l \times n_d}$ and $P_u \in \mathbb{R}^{n_u \times n_d}$ corresponding to the sets $\mathcal{I}_l$ and $\mathcal{I}_u$, respectively, as

$$P_l(i,j) := \begin{cases} 1 & \text{if } j = p_i \quad \text{for} \quad i = 1, 2, \ldots, n_l, \ j = 1, 2, \ldots, n_d \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

$$P_u(i,j) := \begin{cases} 1 & \text{if } j = q_i \quad \text{for} \quad i = 1, 2, \ldots, n_u, \ j = 1, 2, \ldots, n_d \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

The lower and upper bounds on x can be combined with the inequality constraint

matrix D so that (2.1) becomes

$$x^* := \arg\min_x \frac{1}{2} x^T H x + h^T x \quad \text{subject to} \quad Gx \leq g, \quad Fx = f, \tag{2.4}$$

where

$$G := \begin{bmatrix} -P_l \\ P_u \\ D \end{bmatrix}, \quad g := \begin{bmatrix} -x_{\text{lb}} \\ x_{\text{ub}} \\ d \end{bmatrix}, \quad x_{\text{lb}} := \begin{bmatrix} \underline{x}_{p_1} \\ \underline{x}_{p_2} \\ \vdots \\ \underline{x}_{p_{n_l}} \end{bmatrix}, \quad x_{\text{ub}} := \begin{bmatrix} \overline{x}_{q_1} \\ \overline{x}_{q_2} \\ \vdots \\ \overline{x}_{q_{n_u}} \end{bmatrix}. \tag{2.5}$$

For the special case when $H = 0$, (2.4) reduces to a linear program (LP).

To solve the QP problem two approaches are commonly used, namely active set methods [12] and interior point methods (IPMs) [26, 30]. We focus on IPMs because they have polynomial computational complexity, while active set methods have exponential complexity in the worst case.

In this section we review the ideas behind primal-dual IPMs [30]. The KKT conditions of (2.4) are

$$Hx + F^T y + G^T z + h = 0, \tag{2.6a}$$
$$Fx - f = 0, \tag{2.6b}$$
$$Gx - g + s = 0, \tag{2.6c}$$
$$ZS\mathbf{1}_{n_t} = 0, \quad z, s \geq 0, \tag{2.6d}$$

where the number of inequality constraints $n_t := n_l + n_u + n_i$, $y \in \mathbb{R}^{n_e}$ and $z \in \mathbb{R}^{n_t}$ are called dual variables, $s \in \mathbb{R}^{n_t}$ is a vector of slack variables, $\mathbf{1}_{n_t} \in \mathbb{R}^{n_t}$ is a vector of ones and Z and S are diagonal matrices defined by $Z := \text{diag}(z), S := \text{diag}(s)$, whose diagonal elements are the components of z and s, respectively.

In many IPMs the optimal solution is obtained by solving the nonlinear optimality conditions (2.6). The classical algorithm to solve such equations is Newton's method. This is an iterative method in which, at each iteration k, the solution of a linear system of the following form is required to find the search direction:

$$\begin{bmatrix} H & F^T & G^T & 0 \\ F & 0 & 0 & 0 \\ G & 0 & 0 & I \\ 0 & 0 & S^k & Z^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \\ \Delta s^k \end{bmatrix} = - \begin{bmatrix} r_H^k \\ r_F^k \\ r_G^k \\ r_S^k \end{bmatrix}, \tag{2.7}$$

where

$$r_H^k := Hx^k + F^T y^k + G^T z^k + h, \tag{2.8a}$$
$$r_F^k := Fx^k - f, \tag{2.8b}$$
$$r_G^k := Gx^k - g + s^k, \tag{2.8c}$$
$$r_S^k := Z^k S^k \mathbf{1}_{n_t} - \sigma \mu^k \mathbf{1}_{n_t}, \tag{2.8d}$$

$\sigma \in (0, 1)$ is a centering parameter and the duality gap is defined as

$$\mu^k := (z^k)^T s^k / n_t. \tag{2.9}$$

The linear system (2.7) is unsymmetric, but by block elimination can be reduced to a smaller and symmetric system

$$\underbrace{\begin{bmatrix} H & F^T & G^T \\ F & 0 & 0 \\ G & 0 & -W^k \end{bmatrix}}_{\mathcal{A}_1^k} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta z^k \end{bmatrix} = - \underbrace{\begin{bmatrix} r_H^k \\ r_F^k \\ r_L^k \end{bmatrix}}_{b_1^k}, \tag{2.10a}$$

$$\Delta s^k = -(Z^k)^{-1} \left(r_S^k + S^k \Delta z^k \right), \tag{2.10b}$$

where $W^k := (Z^k)^{-1} S^k$ is a diagonal matrix and $r_L^k := r_G^k - (Z^k)^{-1} r_S^k$. System (2.10a) is symmetric indefinite and is often more convenient to solve than (2.7).

A further reduction in the number of unknowns in (2.10a) can be made using another block elimination, which results in

$$\underbrace{\begin{bmatrix} H + G^T (W^k)^{-1} G & F^T \\ F & 0 \end{bmatrix}}_{\mathcal{A}_2^k} \begin{bmatrix} \Delta x^k \\ \Delta y^k \end{bmatrix} = - \underbrace{\begin{bmatrix} r_H^k + G^T (W^k)^{-1} r_L^k \\ r_F^k \end{bmatrix}}_{b_2^k}, \tag{2.11a}$$

$$\Delta z^k = (W^k)^{-1} \left(G \Delta x^k + r_L^k \right). \tag{2.11b}$$

System (2.11a) is also symmetric indefinite and smaller than (2.10a). However, the matrix in (2.11a) involves a double product and may be less sparse. The double product is also a major ingredient in the total cost of an IPM. The symmetric indefinite linear system (2.10a) or (2.11a) can be solved by a direct method such as LDL^T factorization, or by an iterative method such as MINRES.

If the QP problem (2.4) does not contain equality constraints, then (2.11a) reduces to

$$\underbrace{\left(H + G^T (W^k)^{-1} G \right)}_{\mathcal{A}_3^k} \Delta x^k = \underbrace{r_H^k + G^T (W^k)^{-1} r_L^k}_{b_3^k}. \tag{2.12}$$

This positive definite linear system can be solved by a direct method such as Cholesky factorization or by an iterative method such as the conjugate gradient (CG) method.

The product $G^T (W^k)^{-1} G$ in (2.11) and (2.12) can be written as

$$G^T (W^k)^{-1} G = P_l^T (W_l^k)^{-1} P_l + P_u^T (W_u^k)^{-1} P_u + D^T (W_d^k)^{-1} D, \tag{2.13}$$

where

$$W_l^k := \operatorname{diag}(w_1^k, w_2^k, \ldots, w_{n_l}^k), \tag{2.14a}$$

$$W_u^k := \operatorname{diag}(w_{n_l+1}^k, w_{n_l+2}^k, \ldots, w_{n_l+n_u}^k), \tag{2.14b}$$

$$W_d^k := \operatorname{diag}(w_{n_l+n_u+1}^k, w_{n_l+n_u+2}^k, \ldots, w_{n_l+n_u+n_i}^k). \tag{2.14c}$$

The first two parts in (2.13) can be computed by just picking the rows and columns of $(W_l^k)^{-1}$ and $(W_u^k)^{-1}$ according to the sets $\mathcal{I}_l$ and $\mathcal{I}_u$, respectively. Hence, using (2.13) we can reduce the cost of the product. This indicates that if (2.1) has only lower and upper bounds on x then the cost of the product $G^T (W^k)^{-1} G$ in (2.11) or (2.12) can be eliminated.

For the special case when $H = 0$ and there are no equality constraints, only lower and upper bounds on x in (2.1), the matrix $\mathcal{A}_3^k$ in (2.12) becomes diagonal and easy to invert.

2.1. Exact interior point methods. An *exact* IPM solves the linear systems (2.10a), (2.11a) or (2.12) by either a direct method or a suitable iterative method with sufficiently small tolerance. An IPM in which an initial guess satisfies $(x^0, z^0, s^0) \in \{(x, z, s) \mid Fx = f, Gx + s = g, (z, s) > 0\}$ is called a *feasible* IPM. An IPM for which we only require $(z^0, s^0) > 0$ is called an *infeasible* IPM. Many log-barrier [5] and potential reduction [7] methods are feasible IPMs. For such methods, one needs to compute a feasible starting point by a Phase I method [5]. However, it is also possible to modify log-barrier and potential reduction methods to allow for an infeasible starting point; for example, one can introduce slack variables s for the inequality constraints in the primal problem ($Gx \leq g$ is replaced by $Gx - g + s = 0$ and $s \geq 0$ in (2.4)), or adopt a suitably-modified primal-dual formulation [7], followed by an infeasible-start Newton method [5]. An infeasible IPM for a QP is described in Algorithm 1; this is an extension of the infeasible-path-following IPM of [30, p. 110], which was developed for an LP.

The right-hand side of (2.10a), (2.11a) and (2.12) is denoted by b^k in Algorithm 1; and the central path neighbourhood $\mathcal{N}_{-\infty}(\gamma, \beta)$ in Algorithm 1 is defined as [30, p. 109]

$$\mathcal{N}_{-\infty}(\gamma, \beta) := \{(x^k, y^k, z^k, s^k) \mid \|(r_F^k, r_G^k)\| \leq \|(r_F^0, r_G^0)\| \beta \mu^k / \mu^0, \\ (z^k, s^k) > 0,\ z_i s_i \geq \gamma \mu^k,\ i = 1, 2, \ldots, n_t\}, \tag{2.17}$$

where $\gamma \in (0, 1)$ and $\beta \geq 1$ are given parameters.

In (2.17), $\|(r_F^k, r_G^k)\|$ indicates a measure of infeasibility and is uniformly bounded by some multiple of μ^k. Since μ^k is monotonically decreasing, as indicated by (2.16), $\mu^k \to 0$ as $k \to \infty$, hence $\|(r_F^k, r_G^k)\| \to 0$. This indicates that Algorithm 1 guarantees feasibility if a feasible solution exists, i.e. $r_F^k \to 0$ and $r_G^k \to 0$ as $k \to \infty$. However, if no feasible solution exists, then from [30, Thm 9.7] it follows that $\|(r_F^k, r_G^k)\| \to \infty$ as $k \to \infty$. One simple approach that is used in detecting infeasiblity is to terminate the algorithm if $\|(r_F^k, r_G^k)\| > \omega$ for a sufficiently large value of ω. Other practical approaches for detecting infeasibility and terminating the IPM algorithm are given in [15].

If a point lies in $\mathcal{N}_{-\infty}(\gamma, \beta)$, each pairwise product $z_i^k s_i^k$ must be greater than $\gamma \mu^k$. By using a suitably small value of γ we can cover most of the feasible region. The typical value of this parameter is 10^{-3} [30, p. 9]. By choosing a very large value of β, we may reduce the infeasibility at a much faster rate, but on the other hand, it may lead to a very small step length α^k.

2.2. Inexact interior point methods. Direct methods have been extensively used to solve (2.11a) and (2.12) in interior point software, such as MOSEK [22] and OOQP [15]. The Cholesky factorization is usually used for a symmetric positive definite matrix and LDL^T factorization for a symmetric indefinite matrix. For sparse linear systems, sparse direct methods [9] can be applied to minimize fill-ins (during the factorization process, non-zero values are produced in some positions that were initially zero). The ill-conditioning of the matrix is not a big issue for direct methods, as it is with iterative methods. However, the cost of solving large linear systems by direct methods may become excessive in terms of memory and computational time. Iterative methods have an advantage in requiring less memory.

Another disadvantage of direct methods is that it is not possible to terminate the solver early and obtain an approximate solution to the system of equations. For example, when the current iterate is far from the solution, an approximate solution to the linear system might be a "sufficiently good" search direction. Iterative methods,

Algorithm 1 Exact/Inexact Infeasible Interior Point Method

Input:

- H, F, G, f, d
- Initial guess $x^0, y^0, z^0 > 0, s^0 > 0, \gamma \in (0,1), \beta \geq 1, \sigma \in (0,1)$
- Tolerance $\epsilon > 0$

Output: Optimal x

Algorithm:

Set $k = 0$ and compute $\mu^0 := \frac{(z^0)^T s^0}{n_t}$, $e_{tol}^0 := \|b^0\|$, where b^0 is the right hand side of (2.10a), (2.11a) or (2.12) for $k = 0$.
while $\mu^k > \epsilon$ and $e_{tol}^k > \epsilon$ **do**
  if Exact IPM **then**
    Solve (2.10a) or (2.11a) by LDL^T factorization or MINRES. If (2.1) does not contain equality constraints, then (2.12) can be solved by Cholesky factorization or CG. The solution computed at the previous outer iteration is used as an initial guess for MINRES or CG. MINRES or CG is terminated when $\|r^{k,j}\|/\|b^k\| < \epsilon$, where j is the inner iteration count of the MINRES/CG solver, $r^{k,j}$ is defined in (2.19) and ϵ is sufficiently small.
  else if Inexact IPM **then**
    Solve (2.10a) or (2.11a) by MINRES or (2.12) by CG if (2.1) does not contain equality constraints, with relative residual tolerance defined by (2.20). The solution computed at the previous outer iteration is used as an initial guess for MINRES/CG.
  end if
  Choose α^k as the largest value in $(0, 1]$ such that the following conditions hold:

$$(x^k, y^k, z^k, s^k) + \alpha^k(\Delta x^k, \Delta y^k, \Delta z^k, \Delta s^k) \in \mathcal{N}_{-\infty}(\gamma, \beta), \tag{2.15}$$

$$\frac{(z^k + \alpha^k \Delta z^k)(s^k + \alpha^k \Delta s^k)}{n_t} \leq (1 - 0.01\alpha^k)\mu^k. \tag{2.16}$$

  Compute $(x^{k+1}, y^{k+1}, z^{k+1}, s^{k+1}) := (x^k, y^k, z^k, s^k) + \alpha^k(\Delta x^k, \Delta y^k, \Delta z^k, \Delta s^k)$.
  Compute $\mu^{k+1} := \frac{(z^{k+1})^T s^{k+1}}{n_t}$, $e_{tol}^{k+1} := \|b^{k+1}\|$.
  Increment k by 1.
end while

on the other hand, can be terminated early and form the basis of so-called inexact or truncated Newton methods [23].

An IPM that solves (2.10a), (2.11a) or (2.12) only approximately is called an *inexact* interior point method (IIPM); an IIPM is also described in Algorithm 1. In an IIPM the total number of floating-point operations needed to find the solution to (2.1) can be reduced significantly by solving (2.10a), (2.11a) or (2.12) approximately, compared to an exact method [11]. Since direct methods cannot be terminated early, they are not applicable here.

In an IIPM, equations (2.10a), (2.11a) or (2.12) can be written in the form

$$\mathcal{A}^k p^k = b^k \tag{2.18}$$

and solved with an iterative method, such as MINRES or CG, which is terminated

when

$$\frac{\|r^{k,j}\|}{\|b^k\|} \leq \eta^k \text{ where } r^{k,j} := b^k - \mathcal{A}^k p^{k,j} \tag{2.19}$$

for a given $\eta^k < 1$. We refer to the while loop of Algorithm 1 as the outer loop and the loop within the linear solver as the inner loop, so that the superscript j represents the iteration number of the inner loop, while the superscript k represents the iteration number of the outer loop. The nonnegative forcing sequence $\{\eta^k\}$ is used to control the accuracy at each iteration k of an IIPM. Different methods use different choices of forcing sequence, which affect the efficiency of an inexact method. In the numerical results presented in Section 6, we have selected the following decreasing function [11]:

$$\eta^k := \max\left\{\min\left\{\frac{1}{(k+1)^\zeta}, \|b^k\|\right\}, \epsilon\right\}, \tag{2.20}$$

where $\epsilon > 0$ is the given tolerance level. Note that η^k is larger for smaller values of k when x^k is far from the solution of (2.1) and hence fewer iterations may be sufficient to find the search direction. As k increases, $\|b^k\|$ decreases, which implies η^k decreases and p^k becomes closer to the Newton search direction in the later iterations of the IIPM. It is shown in [11] that, under the assumption that the forcing sequence $\{\eta^k\}$ is uniformly less than one, the inexact algorithm is locally convergent. The parameter ζ provides a compromise between the number of inner and outer iterations. As ζ increases the number of inner iterations decreases while outer iterations increase, and vice versa. Therefore, an optimum value of ζ can be selected for a specific problem in which the total computational cost of Algorithm 1 is minimum for a given set of initial conditions.

An alternative termination criterion for the linear solver could be to use $\|r^{k,j}\| \leq \eta^k(\|\mathcal{A}^k\|\|p^{k,j}\| + \|b^k\|)$, which will be reached sooner than the one in (2.19), potentially decreasing the overall amount of computational effort of the IPM. However, the implications of the use of this termination criterion in terms of local convergence properties could be a topic of further research.

Note that the final output of an inexact IPM is within the same tolerance of the solution to (2.1) as the output of an exact IPM. The only difference is in the early iterations of the IPM; in later iterations the (x^k, y^k, z^k, s^k) from the inexact IPM converges to the (x^k, y^k, z^k, s^k) of the exact IPM. The only part where an approximate solution to the linear system is obtained is in the early iterations.

3. New approximate method for computing the search direction. The solution of (2.10a), (2.11a) or (2.12) is a key part of Algorithm 1. It is well-known that the condition number of the matrices $\mathcal{A}_1^k$, $\mathcal{A}_2^k$ and $\mathcal{A}_3^k$ increases as k increases [30, p. 217] and iterative methods do not perform well with ill-conditioned systems. Therefore, in the IPM literature, the main focus in solving such linear systems is often on direct methods. However, in the last decade, there has been an increasing trend in using iterative methods with suitable preconditioners, with promising results [2,3,20].

In this section we propose solving a suitably defined approximation of (2.10a), where the condition number of the resulting approximate linear system is improved, thereby allowing the use of iterative methods. Furthermore, this approximate linear system is smaller than (2.10a), and hence can be solved more efficiently by direct or iterative methods.

We start by introducing the concept of a δ-*active set* for a given scalar $\delta > 0$. We define the δ-active set $\mathcal{N}_A^k(\delta)$ at iteration k as

$$\mathcal{N}_A^k(\delta) := \{i \in \mathcal{N} \mid 0 < w_i^k < \delta\}, \text{ where } w_i^k := \frac{s_i^k}{z_i^k} \tag{3.1}$$

and the δ-inactive set at iteration k as $\mathcal{N}_I^k(\delta) := \mathcal{N} \backslash \mathcal{N}_A^k(\delta)$. In the first IPM iteration, the value of δ is usually selected sufficiently large such that all inequality constraints are δ-active, i.e. $\delta > \frac{s_i^0}{z_i^0}$ for each i. The selection of δ is discussed further in Section 3.3. The above sets, though related, should not be confused with the *active set* and *inactive set* at a solution to (2.4).

Let the number of δ-active constraints at iteration k be denoted by

$$n_a^k := \text{card}(\mathcal{N}_A^k(\delta)) \tag{3.2}$$

and consider permuting and splitting (2.10a) according to the δ-active and δ-inactive constraints as

$$\underbrace{\left[\begin{array}{ccc|c} H & F^T & (G_1^k)^T & (G_2^k)^T \\ F & 0 & 0 & 0 \\ G_1^k & 0 & -W_1^k & 0 \\ \hline G_2^k & 0 & 0 & -W_2^k \end{array}\right]}_{\mathcal{A}_4^k} \left[\begin{array}{c} \Delta x^k \\ \Delta y^k \\ \Delta z_1^k \\ \hline \Delta z_2^k \end{array}\right] = \left[\begin{array}{c} r_H^k \\ r_F^k \\ r_{L_1}^k \\ \hline r_{L_2}^k \end{array}\right], \tag{3.3}$$

where

$$\|W_1^k\|_\infty < \delta, \quad \|(W_2^k)^{-1}\|_\infty \leq \delta^{-1}, \tag{3.4}$$

$$\left[\begin{array}{c} G_1^k \\ G_2^k \end{array}\right] := U^k G, \quad \left[\begin{array}{c} \Delta z_1^k \\ \Delta z_2^k \end{array}\right] := U^k \Delta z^k, \quad \left[\begin{array}{c} r_{L_1}^k \\ r_{L_2}^k \end{array}\right] := U^k r_L^k, \tag{3.5}$$

$W_1^k \in \mathbb{R}^{n_a^k \times n_a^k}$, $W_2^k \in \mathbb{R}^{(n_t - n_a^k) \times (n_t - n_a^k)}$, $U^k \in \mathbb{R}^{n_t \times n_t}$ is a suitably-defined permutation matrix such that (3.4) holds, and the infinity norm of a matrix $K \in \mathbb{R}^{m \times n}$ is defined as $\|K\|_\infty := \max_{1 \leq i \leq m} \sum_{j=1}^n |k_{ij}|$, where k_{ij} is the ij^{th} element of K. Note that $\|(W_2^k)^{-1}\| \leq \delta^{-1}$ is equivalent to requiring that the smallest component of W_2^k be greater than or equal to δ.

The main idea is to omit the matrix $(G_2^k)^T$ from (3.3) and to solve the resulting block–triangular system (either accurately or approximately) to obtain an approximate search direction. For simplicity, we write (3.3) as

$$\left[\begin{array}{cc} A_1^k & (A_2^k)^T \\ A_2^k & -W_2^k \end{array}\right] \left[\begin{array}{c} p_1^k \\ p_2^k \end{array}\right] = \left[\begin{array}{c} b_1^k \\ b_2^k \end{array}\right], \tag{3.6}$$

where

$$A_1^k := \left[\begin{array}{ccc} H & F^T & (G_1^k)^T \\ F & 0 & 0 \\ G_1^k & 0 & -W_1^k \end{array}\right], \quad (A_2^k)^T := \left[\begin{array}{c} (G_2^k)^T \\ 0 \\ 0 \end{array}\right], \tag{3.7a}$$

$$p_1^k := \left[\begin{array}{c} \Delta x^k \\ \Delta y^k \\ \Delta z_1^k \end{array}\right], \quad b_1^k := \left[\begin{array}{c} r_H^k \\ r_F^k \\ r_{L_1}^k \end{array}\right], \quad p_2^k := \Delta z_2^k, \quad b_2^k := r_{L_2}^k. \tag{3.7b}$$

We are now in a position to state the main point of this section. In the later IPM iterations, the values of the diagonal matrix W_2^k become very large and hence $\|(W_2^k)^{-1}\|_\infty \to 0$. Therefore, if δ is chosen to be sufficiently large at the beginning, then $p_2^k = (W_2^k)^{-1}(A_2^k p_1^k - b_2^k) \approx 0$ at later iterations, and hence $A_1^k p_1^k \approx b_1^k$. We therefore propose that, rather than solving (2.10a) or (3.6), we find an approximate solution of (3.6) by solving

$$A_1^k \hat{p}_1^k = b_1^k, \tag{3.8a}$$
$$W_2^k \hat{p}_2^k = A_2^k \hat{p}_1^k - b_2^k. \tag{3.8b}$$

Since W_2^k is diagonal, the main task is to solve the smaller linear system (3.8a). This can yield significant computational savings if $n_a^k \ll n_t$, as is the case in later iterations for many practical applications.

A similar idea to the above was proposed in [13], where they set $\delta = 1$ and presented a numerically stable method for solving the partitioned system (3.3) inexactly using the iterative solver QMR. We take this idea a step further and propose solving instead the smaller system (3.8a), which can be solved exactly or inexactly with a direct or iterative solver.

In Section 3.4 we present some results that suggest that the distribution of the eigenvalues of A_1^k is more favorable than the distribution of eigenvalues for A_4^k in (3.3). This implies that if we wish to solve (3.8a) with an iterative method such as MINRES, then the rate of convergence is improved compared to solving (3.3) with an iterative method, and lower precision arithmetic may be used.

It is important to note that if a constraint is δ-inactive it is not omitted from the computation of the search direction. The components of w^k are used to define how the linear system (3.3) is permuted in order to construct the approximate system (3.8). Constraints that are δ-active or δ-inactive are considered in (3.8a) or (3.8b), respectively. Also, the constraint classification based on δ is made at each IPM iteration; hence the classification of a constraint can change at the early iterations and will only settle into a final classification at later iterations.

3.1. Reducing the size of the approximate linear system (3.8a). To reduce the number of unknowns in (3.8a), we may perform block elimination and solve

$$\left(U_1^k V (U_1^k)^T + W_1^k\right) \Delta\hat{z}_1^k = U_1^k V_0 r_{\bar{H}}^k - r_{L_1}^k, \tag{3.9}$$

where

$$\bar{H} := \begin{bmatrix} H & F^T \\ F & 0 \end{bmatrix}, \quad r_{\bar{H}}^k := \begin{bmatrix} r_H^k \\ r_F^k \end{bmatrix}, \quad U^k =: \begin{bmatrix} U_1^k \\ U_2^k \end{bmatrix}, \tag{3.10a}$$

$$V := \bar{G}\bar{H}^{-1}(\bar{G})^T, \quad V_0 := \bar{G}\bar{H}^{-1}, \quad \bar{G} := [G\ 0], \tag{3.10b}$$

and $U_1^k \in \mathbb{R}^{n_a^k \times n_t}$, provided $\bar{H}$ is nonsingular. Note that if $\mathcal{N}_A^k(\delta) = \emptyset$, then $A_1^k = \bar{H}$. If $F = 0$, i.e. there is no equality constraint in (2.1), then $\bar{H} = H$.

The remaining part of the solution of (3.8) is calculated from

$$\bar{H} \begin{bmatrix} \Delta\hat{x}^k \\ \Delta\hat{y}^k \end{bmatrix} = \begin{bmatrix} r_H^k - (G_1^k)^T \Delta\hat{z}_1^k \\ r_F^k \end{bmatrix} \tag{3.11}$$

and

$$W_2^k \Delta\hat{z}_2^k = G_2^k \Delta\hat{x}_k - r_{L_2}^k. \tag{3.12}$$

If $\bar{H}$ is non-singular, V and V_0 can be computed by factoring $\bar{H}$ using LDL^T or Cholesky factorization if $\bar{H}$ is indefinite or positive definite, respectively. Since V and V_0 are independent of the IPM iteration number k, they can be computed outside the main loop. The matrix $U_1^k V (U_1^k)^T$ can be formed by picking the rows and columns of V according to U_1^k, and $U_1^k V_0$ can be computed in a similar fashion. Hence, no matrix product is involved in computing the matrix $U_1^k V (U_1^k)^T$. Inside the main loop one can solve (3.11) and compute $U_1^k V_0 r_{\bar{H}}^k$ by backward and forward substitution.

Recall that if H is positive semidefinite and F has full row rank, then a necessary and sufficient condition for $\bar{H}$ to be non-singular is $\ker(H) \cap \ker(F) = \{0\}$, where $\ker(\cdot)$ denotes the kernel of a matrix [2, Thm 3.2]. For a positive definite H, $\bar{H}$ would be non-singular if and only if F has full row rank [2]. If not, (2.1) can always be converted to an equivalent problem in which F does have full row rank [30]. For the class of control problems considered in Section 6 one can show that F has full row rank and that $\bar{H}$ is non-singular.

If H, F and G are dense, we prefer to compute V and V_0 as above if possible. However, if H, F and G are sparse, this approach may destroy sparsity. To exploit sparsity, one might start by computing a sparse factorization of $\bar{H}$ if possible. Alternatively, one could solve (3.8) directly instead of solving (3.9) and (3.11), especially if $\bar{H}$ is singular.

A δ-active infeasible IPM is summarized in Algorithm 2 for the case when (3.9) and (3.11) are solved. The modifications for the case when (3.8) is solved instead are straightforward.

Lines 1–3 in Algorithm 2 are outside the *while* loop and therefore do not add much to the total computational cost of the algorithm. At line 5 the δ-active set can be computed by just comparing the elements of w_i^k with δ. The system matrix in (3.9) is computed as explained earlier. Lines 7–10 are straightforward. The criteria for selection of step length α^k in line 11 are the same as in [30, p. 110].

3.2. Error in the solution of (3.8). The approximation of (3.6) by (3.8) reduces the computational cost at the expense of introducing an error. An upper bound on the error is estimated in the following result.

PROPOSITION 3.1. *Let*

$$e_1^k := \hat{p}_1^k - p_1^k,$$
$$e_2^k := \hat{p}_2^k - p_2^k$$

be the error in the solution of (3.6). Then

$$\|e_1^k\|_\infty \leq \delta^{-1} \left(c_1^k \|b_1^k\|_\infty + c_2^k \|b_2^k\|_\infty \right) + O(\delta^{-2}), \tag{3.13a}$$
$$\|e_2^k\|_\infty \leq \delta^{-1} \left(\|e_1^k\|_\infty \|A_2^k\|_\infty \right) + O(\delta^{-2}), \tag{3.13b}$$

where

$$c_1^k := (c_3^k)^2 \|G^T\|_\infty \|G\|_\infty, \tag{3.14a}$$
$$c_2^k := c_3^k \|G^T\|_\infty, \tag{3.14b}$$
$$c_3^k := \frac{\|G\|_\infty}{\|\bar{H}\|_\infty} \kappa_\infty(A_1^k), \tag{3.14c}$$

and $\kappa_\infty(A_1^k)$ is the condition number of A_1^k.

Algorithm 2 New δ-active Exact/Inexact Infeasible Interior Point Method

Input:

- H, F, G, f, d
- Initial guess $x^0, y^0, z^0 > 0, s^0 > 0, \beta \geq 1, k_{\max}$
- Select $\delta, \alpha_{\min}$
- Tolerance $\epsilon > 0$

Output: Optimal x.

Algorithm:

1: Compute a factorization of $\bar{H}$, defined in (3.10a).
2: Compute V and V_0, defined in (3.10b).
3: Set $k = 0$ and compute $\mu^0 := \frac{(z^0)^T s^0}{n_t}$, $e_{tol}^0 := \|b^0\|$.
4: **while** $\mu^k > \epsilon$ and $e_{tol}^k > \epsilon$ **do**
5: Compute the δ-active set $\mathcal{N}_A^k(\delta)$.
6: Solve (3.9) for $\Delta\hat{z}_1^k$. In case an inexact IPM with an iterative linear solver is to be implemented, the relative residual tolerance defined by (2.19) can be used.
7: Compute $\Delta\hat{x}^k$ and $\Delta\hat{y}^k$ from (3.11) using forward and backward substitution.
8: Compute $\Delta\hat{z}_2^k$ from (3.12).
9: Set $\Delta x^k := \Delta\hat{x}^k, \Delta y^k := \Delta\hat{y}^k, \Delta z^k := (U^k)^T \begin{bmatrix} \Delta\hat{z}_1^k \\ \Delta\hat{z}_2^k \end{bmatrix}$.
10: Compute Δs^k from (2.10b).
11: Choose α^k as the largest value in $(0, 1]$ such that following conditions hold:

$$(x^k, y^k, z^k, s^k) + \alpha^k(\Delta x^k, \Delta y^k, \Delta z^k, \Delta s^k) \in \mathcal{N}_{-\infty}(\gamma, \beta),$$
$$\frac{(z^k + \alpha^k \Delta z^k)(s^k + \alpha^k \Delta s^k)}{n_t} \leq (1 - 0.01\alpha^k)\mu^k.$$

12: **if** $\alpha^k < \alpha_{\min}$ or $k > k_{\max}$ **then**
13: Replace δ by 1.5δ and go to Step 5.
14: **end if**
15: Compute $(x^{k+1}, y^{k+1}, z^{k+1}, s^{k+1}) := (x^k, y^k, z^k, s^k) + \alpha^k(\Delta x^k, \Delta y^k, \Delta z^k, \Delta s^k)$.
16: Compute $\mu^{k+1} := \frac{(z^{k+1})^T s^{k+1}}{n_t}$, $e_{tol}^{k+1} := \|b^{k+1}\|$.
17: Increment k by 1.
18: **end while**

Proof. From (3.6),

$$A_1^k p_1^k + (A_2^k)^T p_2^k = b_1^k, \tag{3.15a}$$
$$A_2^k p_1^k - W_2^k p_2^k = b_2^k, \tag{3.15b}$$

From (3.15b),

$$p_2^k = \delta^{-1}(\bar{W}_2^k)^{-1}(A_2^k p_1^k - b_2^k) \tag{3.16}$$

where $(\bar{W}_2^k)^{-1} := \delta(W_2^k)^{-1}$ and $\|(\bar{W}_2^k)^{-1}\| \leq 1$. From (3.8b) and (3.9), we get

$$e_2^k = \delta^{-1}(\bar{W}_2^k)^{-1} A_2^k e_1^k. \tag{3.17}$$

Replacing p_2^k in (3.15a), we get

$$p_1^k = \left(A_1^k + \delta^{-1} X_1^k\right)^{-1} \left(b_1^k + \delta^{-1} X_2^k b_2^k\right) \tag{3.18}$$

where

$$X_1^k := (A_2^k)^T(\bar{W}_2^k)^{-1}A_2^k \text{ and } X_2^k := (A_2^k)^T(\bar{W}_2^k)^{-1}. \tag{3.19}$$

From (3.7a), we get

$$X_1^k = \begin{bmatrix} (G_2^k)^T(\bar{W}_2^k)^{-1}G_2^k & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \; X_2^k = \begin{bmatrix} (G_2^k)^T(\bar{W}_2^k)^{-1} \\ 0 \\ 0 \end{bmatrix}. \tag{3.20}$$

For small ε and $K, X \in \mathbb{R}^{n\times n}$, $(K+\varepsilon X) = K(I+\varepsilon K^{-1}X)$ and so $(K+\varepsilon X)^{-1} = \left(I - \varepsilon K^{-1}X + O(\varepsilon^2)\right)K^{-1}$. From (3.18),

$$p_1^k = (A_1^k)^{-1}b_1^k + \delta^{-1}\left(C_1^k b_1^k + C_2^k b_2^k\right) + O(\delta^{-2}), \tag{3.21}$$

where

$$C_1^k := -(A_1^k)^{-1}X_1^k(A_1^k)^{-1} \text{ and } C_2^k := (A_1^k)^{-1}X_2^k. \tag{3.22}$$

From (3.8), (3.21) and (3.9), we get

$$e_1^k = \delta^{-1}\left(C_1^k b_1^k + C_2^k b_2^k\right) + O(\delta^{-2}). \tag{3.23}$$

Taking the infinity norm of (3.23) and (3.17), we get (3.13). ∎

Note that $\|b^k\| < \epsilon$ at the termination of the IPM. By choosing appropriate values of ϵ, δ, and k it is possible to make e^k arbitrarily small. As the iteration number k of the new IPM increases, the values of the diagonal matrix W_2^k corresponding to the δ-inactive constraints become very large. Therefore, $w_i^k \to \infty$ for the constraints that are inactive at the solution and hence $\|(W_2^k)^{-1}\|_\infty \to 0$. From (3.19), we get $X_1^k \to 0$ and $X_2^k \to 0$, and hence (3.18) implies $p_1^k \to (A_1^k)^{-1}b_1^k$, which is equal to $\hat{p}_1^k$. Therefore we get $e_1^k \to 0$. From (3.17) we get $e_2^k \to 0$. This indicates that $e_1^k \to 0$ and $e_2^k \to 0$ for any non-zero positive value of δ.

3.3. Selection of δ. The computational cost of Algorithm 2 and the error in (3.8) clearly depends upon δ. Good values for δ and the other parameters are likely to be problem-dependent and could be determined experimentally. In practice, we have found that a reasonable lower bound for δ is such that all inequality constraints are δ-active for $k = 0$; i.e. $\mathcal{N}_A^0(\delta) = \mathcal{N}$.

In later iterations, if δ is too small it might lead to blocking of the search direction, which means that α^k becomes too small and the next iterate will be close to the current iterate. The value of δ can then be increased using any sensible heuristic, such as the one in lines 12–14 of Algorithm 2. A simple approach is to multiply it by 1.5, for example, as in Algorithm 2. Any small value of $\alpha_{\min}$ can be used; we have used $\alpha_{\min} := 10^{-4}$ in our numerical examples. The maximum number of IPM iterations required for the solution of a QP is usually in the range of 10–20. Therefore, the value of $k_{\max}$ can be selected to be 20, for example.

In many applications, such as in optimal control, we need to solve on-line a sequence of QPs in which the data of each QP is slightly perturbed from the previous one. For example, suppose we have a set of QPs in which each QP has the form (2.4) and only f changes from one QP to another, as in Section 6. Let $\mathcal{F}$ be the set of vectors f for which a solution to (2.4) exists. During the off-line design phase one could use, for example, Monte Carlo methods to estimate upper and/or lower bounds for $\delta^* := \arg\inf_\delta \sup_{f\in\mathcal{F}} t(\delta, f)$, where $t(\delta, f)$ is the time taken by Algorithm 2 to terminate for a given δ and f. During the on-line implementation phase, the value of δ is then fixed to be slightly larger than the estimated bound on δ^*.

3.4. Results on the spectrum of the matrix A_1^k in (3.8a). As the iteration number k of Algorithm 1 increases, the values corresponding to the δ-inactive constraints of the diagonal matrix W^k become very large. As a result, the condition number of $\mathcal{A}_4^k$ in (3.3) becomes very large. We claim that the distribution of eigenvalues of the matrix in (3.8a) might be more favorable than the distribution for the matrix in (3.3) when considered from the perspective of the numerical stability of a linear solver. We now proceed to provide results supporting this claim.

PROPOSITION 3.2. *Let $\lambda_{\min}(A_1^k), \lambda_{\max}(A_1^k)$ be the minimum and maximum eigenvalues of A_1^k, respectively, then*

$$\lambda_{\min}(A_1^k) \geq \min\left\{h_{\min} - \bar{r}, -\|F\|_\infty, -\delta - \|G\|_\infty\right\}, \tag{3.24a}$$
$$\lambda_{\max}(A_1^k) \leq \max\left\{h_{\max} + \bar{r}, \|F\|_\infty, \|G\|_\infty\right\}, \tag{3.24b}$$

where

$$h_{\min} := \min\{h_{ii} \mid i = 1, 2, \ldots, n_d\}, \tag{3.25a}$$
$$h_{\max} := \max\{h_{ii} \mid i = 1, 2, \ldots, n_d\}, \tag{3.25b}$$
$$\bar{r} := \|H\|_\infty + \|F^T\|_\infty + \|G^T\|_\infty. \tag{3.25c}$$

and h_{ii} is the ii^{th} element of H.

Proof. According to Gershgorin's theorem [16, Thm 8.1.3],

$$\operatorname{spec}(A_1^k) \subseteq \bigcup_{i=1}^{n_d+n_e+n_a^k} [a_{ii} - r_i,\ a_{ii} + r_i], \tag{3.26}$$

where $\operatorname{spec}(A_1^k)$ is the set of eigenvalues of A_1^k, a_{ii} is the ii^{th} element of A_1^k, and

$$r_i := \sum_{j=1,\ j\neq i}^{n_d+n_e+n_a^k} |a_{ij}|. \tag{3.27}$$

From (3.7a) and (3.27), we get

$$r_i \leq \|H\|_\infty + \|F^T\|_\infty + \|G^T\|_\infty \ \ \forall i \in \{1, 2, \ldots, n_d\}, \tag{3.28a}$$
$$r_i \leq \|F\|_\infty \ \ \forall i \in \{n_d + 1, n_d + 2, \ldots, n_d + n_e\}, \tag{3.28b}$$
$$r_i \leq \|G\|_\infty \ \ \forall i \in \{n_d + n_e + 1, n_d + n_e + 2, \ldots, n_d + n_e + n_a^k\}. \tag{3.28c}$$

Recalling that the entries of the diagonal matrix $-W_1^k$ lie between $-\delta < 0$ and 0, it follows from (3.7a), (3.25), (3.26) and (3.28) that

$$\begin{aligned}\operatorname{spec}(A_1^k) \subseteq [&\min\{h_{\min} - \bar{r}, -\|F\|_\infty, -\delta - \|G\|_\infty\}, \\ &\max\{h_{\max} + \bar{r}, \|F\|_\infty, \|G\|_\infty\}]\end{aligned} \tag{3.29}$$

This implies (3.24), which completes the proof. ∎

Note that the upper bound in (3.24b) is not a function of δ. Using the same procedure as in the proof one can also show that the largest eigenvalue of the matrix $\mathcal{A}_4^k$ is bounded above by $\max\{h_{\max} + \bar{r}, \|F\|_\infty, \|G\|_\infty\}$. However, it is not obvious whether one can obtain a bound on the smallest eigenvalue of $\mathcal{A}_4^k$ using the same procedure.

Let $\lambda_i(K)$ denote the i^{th} largest eigenvalue of a symmetric matrix $K \in \mathbb{R}^{n\times n}$ so that $\lambda_n(K) \leq \cdots \leq \lambda_2(K) \leq \lambda_1(K)$ and $\lambda_{\min}(K) := \lambda_n(K),\ \lambda_{\max}(K) := \lambda_1(K)$. In

addition, let the notation K_r denote the submatrix of K containing the first r rows and r columns. According to this notation and using (3.3) and (3.7a), we get

$$A_1^k = (\mathcal{A}_4^k)_{n_c^k}, \quad n_c^k := n_d + n_e + n_a^k. \tag{3.30}$$

PROPOSITION 3.3. *Let $\lambda_i(A_1^k)$ be the i^{th} eigenvalue of the symmetric matrix A_1^k. Then*

$$\lambda_{i+1}\left((\mathcal{A}_4^k)_{n_c^k+1}\right) \leq \lambda_i(A_1^k) \leq \lambda_i\left((\mathcal{A}_4^k)_{n_c^k+1}\right), \tag{3.31}$$

for all $n_a^k < n_t$ and $i \in \{1, 2, \ldots, n_c^k\}$.

Proof. According to the interlacing property [16, Thm 8.1.7], if $K \in \mathbb{R}^{n\times n}$ is a symmetric matrix, then

$$\lambda_{r+1}(K_{r+1}) \leq \lambda_r(K_r) \leq \lambda_r(K_{r+1}) \leq \cdots \leq \lambda_2(K_{r+1}) \leq \lambda_1(K_r) \leq \lambda_1(K_{r+1}) \tag{3.32}$$

for $r = 1, 2, \ldots, n-1$. From (3.30) and (3.32) with $K_r := A_1^k$ and $K_{r+1} := (\mathcal{A}_4^k)_{n_c^k+1}$, we get (3.31). ∎

From Proposition 3.3 we can deduce the following, which tells us how the minimum and maximum eigenvalues of A_1^k relate to the eigenvalues of $\mathcal{A}_4^k$.

COROLLARY 3.4. *For the matrices A_1^k and $\mathcal{A}_4^k$,*

$$\lambda_{\min}(\mathcal{A}_4^k) \leq \lambda_{\min}\left((\mathcal{A}_4^k)_{n_d+n_e+n_t-1}\right) \leq \ldots \leq \lambda_{\min}\left((\mathcal{A}_4^k)_{n_c^k+1}\right) \leq \lambda_{\min}(A_1^k), \tag{3.33}$$

$$\lambda_{\max}(A_1^k) \leq \lambda_{\max}\left((\mathcal{A}_4^k)_{n_c^k+1}\right) \leq \ldots \leq \lambda_{\max}\left((\mathcal{A}_4^k)_{n_d+n_e+n_t-1}\right) \leq \lambda_{\max}(\mathcal{A}_4^k). \tag{3.34}$$

Proof. Using Proposition 3.3 with $i = n_c^k$ we get $\lambda_{\min}\left((\mathcal{A}_4^k)_{n_c^k+1}\right) \leq \lambda_{\min}(A_1^k)$. Using the interlacing property (3.32) with $K := \mathcal{A}_4^k$, $r := n_c^k+1, n_c^k+2, \ldots, n_d+n_e+n_t$ we get

$$\begin{aligned}
\lambda_{\min}\left((\mathcal{A}_4^k)_{n_c^k+2}\right) &\leq \lambda_{\min}\left((\mathcal{A}_4^k)_{n_c^k+1}\right), \\
&\vdots \\
\lambda_{\min}(\mathcal{A}_4^k) &\leq \lambda_{\min}\left((\mathcal{A}_4^k)_{n_d+n_e+n_t-1}\right),
\end{aligned}$$

which results in (3.33). Similarly, using Proposition 3.3 with $i = 1$ we get $\lambda_{\max}(A_1^k) \leq \lambda_{\max}\left((\mathcal{A}_4^k)_{n_c^k+1}\right)$. Using the interlacing property (3.32) with $K := \mathcal{A}_4^k$, $r := n_c^k + 1, n_c^k + 2, \ldots, n_d + n_e + n_t$ we get

$$\begin{aligned}
\lambda_{\max}\left((\mathcal{A}_4^k)_{n_c^k+1}\right) &\leq \lambda_{\max}\left((\mathcal{A}_4^k)_{n_c^k+2}\right), \\
&\vdots \\
\lambda_{\max}\left((\mathcal{A}_4^k)_{n_d+n_e+n_t-1}\right) &\leq \lambda_{\max}(\mathcal{A}_4^k),
\end{aligned}$$

which results in (3.34). ∎

It is observed from numerical experiments that ill-conditioning of $\mathcal{A}_4^k$ at later iterations arises mainly from the increase in the spectral radius of $\mathcal{A}_4^k$. Proposition 3.2 gives bounds on the minimum and maximum eigenvalues of A_1^k as functions of the infinity norms of H, F, G and δ only; note that these bounds are not functions of the iteration count k. Proposition 3.3 and Corollary 3.4 indicate that the minimum and maximum eigenvalues of A_1^k are bounded below and above by the minimum and

maximum eigenvalues of a submatrix of $\mathcal{A}_4^k$, respectively. These results do not give any bound on the distance of the eigenvalues to zero, but show that the spectrum of A_1^k lies within the spectrum of $\mathcal{A}_4^k$. This indicates that the eigenvalues of A_1^k are less scattered than the eigenvalues of $\mathcal{A}_4^k$. Less scattering of eigenvalues results in fewer iterations in the MINRES method, because that number depends upon the distribution of eigenvalues [18, pp. 119–120]. Propositions 3.2, 3.3 and Corollary 3.4 together suggest that the distribution of eigenvalues of the matrix in (3.8a) might be more favorable compared to the spectrum of the matrix in (3.6), and numerical results in Section 6 support this.

4. Preconditioner for (3.9). Preconditioners are generally used in iterative linear solvers, such as CG or MINRES, to enhance the rate of convergence. Before proceeding, we present two new results that are of general interest and will be used to derive a preconditioner and a limit on the number of MINRES or CG iterations when solving (3.9). The following is an extension of Theorem 10.2.5 in [16], which covers CG, to MINRES. The proof is included here for completeness.

PROPOSITION 4.1. *Let* $\mathsf{A} = I + \Delta$ *be a symmetric matrix of size* $n_p \times n_p$ *with* $\text{rank}(\Delta) = q < n_p$. *When solving a linear system* $\mathsf{A}\,p = \mathsf{b}$, *MINRES will terminate in at most* $q + 1$ *iterations.*

Proof. Let p^j and $r^j := \mathsf{b} - \mathsf{A}p^j$ be the value of p and the residual at the j^{th} iteration. If $\mathsf{A} = I + \Delta$ with $\text{rank}(\Delta) = q$, then A will have an eigenvalue at 1 of algebraic multiplicity $n_p - q$. From [18, Eq. 3.7], an upper bound on $\|r^j\|$ can be written as

$$\|r^j\|/\|r^0\| \leq \min_{p_j} \max_{i=1,\ldots,n_p} |p_j(\lambda_i)|\,, \tag{4.1}$$

where $p_j(\lambda_i)$ is a polynomial of degree j and λ_i is the i^{th} eigenvalue of A. Let $p_j(\lambda_i) = 1 + a_1\lambda_i + \cdots + a_j{\lambda_i}^j$, where $a_1, \ldots, a_j$ are unknown coefficients. Substituting this into (4.1), we get

$$\|r_j\|/\|r_0\| \leq \min_{a_1,\ldots,a_j} \max\Big\{|1 + a_1 + \cdots + a_j|, |1 + a_1\lambda_1 + \cdots + a_j\lambda_1^j|, \ldots, \\ |1 + a_1\lambda_q + \cdots + a_j\lambda_q^j|\Big\}\,. \tag{4.2}$$

When $j = q + 1$, the upper bound in (4.2) is zero, because $a_1, \ldots, a_k$ can be uniquely determined by equating each polynomial in (4.2) to zero. Therefore, MINRES will terminate in at most $q + 1$ iterations when solving $\mathsf{A}\,p = \mathsf{b}$. ∎

COROLLARY 4.2. *Let* $(\mathsf{A} + \Delta)$ *be a symmetric matrix of size* $n_p \times n_p$, *where* A *is symmetric and positive definite and* $\text{rank}(\Delta) = q < n_p$. *MINRES with preconditioner* A, *when solving a linear system* $(\mathsf{A} + \Delta)p = \mathsf{b}$, *will terminate in at most* $q + 1$ *iterations.*

The problem of finding the solution of the perturbed system $(\mathsf{A} + \Delta)p = \mathsf{b}$, using the fact that the factorization of A is available in advance, is known as the low-rank update problem. This type of problem arises in many applications of optimization where repeated solution of the linear system is required with a low-rank change in the system matrix. The computational cost of solving the linear system $(\mathsf{A} + \Delta)p = \mathsf{b}$ can be significantly reduced by using low-rank update methods if the rank of the perturbation matrix Δ is sufficiently small.

In the literature, two methods are usually used for low-rank update problems [16]. These are the Sherman-Morrison-Woodbury (SMW) formula and 'update of Cholesky factorization' method. For dense matrices, the number of flops required to solve a

linear system by an iterative method based on Corollary 4.2 is $\mathcal{O}(q\,n_p^2)$. The cost with the SMW formula and Cholesky update is also $\mathcal{O}(q\,n_p^2)$.

For sparse matrices, the number of non-zero elements in $(\mathsf{A}+\Delta)$ increases after the update of the Cholesky factorization, which requires more memory. However, an iterative method based on Corollary 4.2 requires one to store only the resultant vector of the matrix-vector product and is therefore a better option for sparse matrices. For banded matrices, the number of flops for an iterative method based on Corollary 4.2 is $\mathcal{O}(b_w q n_p)$, where b_w is the bandwidth of $(\mathsf{A}+\Delta)$. The SMW formula and Cholesky update also requires $\mathcal{O}(b_w q n_p)$ flops. However, as discussed in Section 1, a method based on an iterative solver is more attractive from a hardware point of view than the SMW formula or Cholesky update.

4.1. A diagonal preconditioner. The size of the linear system (3.9) is $n_a^k \times n_a^k$, so in exact precision MINRES will terminate in at most n_a^k iterations. Note that n_a^k may be more than the number of decision variables n_d, in general. The following fact will be used to design a preconditioner that would ensure that MINRES terminates in at most n_d iterations if $n_d < n_a^k$.

LEMMA 4.3. *The rank of $U_1^k V (U_1^k)^T$ is less than or equal to $\min\{n_a^k, n_d\}$.*

Proof. Since $V = GH_1G^T$, we have $\text{rank}(U_1^k V (U_1^k)^T) = \text{rank}(U_1^k G H_1 G^T (U_1^k)^T) \leq \min\big\{\text{rank}(U_1^k), \text{rank}(H_1^k), \text{rank}(G)\big\}$. Since $\text{rank}(U_1^k) = n_a^k$, $\text{rank}(H_1^k) \leq n_d$ and $\text{rank}(G) \leq \min\{n_d, n_t\}$, it follows that $\text{rank}(U_1^k V (U_1^k)^T) \leq \min\{n_d, n_a^k\}$. ∎

COROLLARY 4.4. *MINRES with preconditioner $\mathcal{P}_1^k := W_1^k$ will terminate in at most $\min\{n_a^k, n_d\}$ iterations when solving (3.9).*

Proof. We know from Lemma 4.3 that $\text{rank}(U_1^k V (U_1^k)^T) \leq \min\{n_a^k, n_d\}$. Therefore, from Corollary 4.2 it follows that MINRES with preconditioner $\mathcal{P}_1^k$ will terminate in at most $\min\{n_a^k, n_d\}$ iterations on system (3.9). ∎

4.2. Another preconditioner. If $n_a^k > n_d$, the preconditioner $\mathcal{P}_1^k$ will be quite effective because it guarantees that the number of MINRES iterations does not exceed n_d. However, if $n_a^k \leq n_d$, the preconditioner $\mathcal{P}_1^k$ might no longer be effective. Furthermore, the diagonals of W_1^k become very small as k increases. In this situation, we propose to use another preconditioner as described next.

In practice, when $n_a^k < n_d$, it can be effective to use the block-diagonal preconditioner $\mathcal{P}_2^k := U_1^k \widetilde{V} (U_1^k)^T + W_1^k$, where $\widetilde{V}$ is a positive semidefinite, block-diagonal approximation of V. Since positive definiteness is a requirement for a preconditioner with MINRES [18, pp. 119–120], a positive semidefinite $\widetilde{V}$ guarantees that $\mathcal{P}_2^k \succ 0$ because $W_1^k \succ 0$. Furthermore, each block-diagonal matrix of $\mathcal{P}_2^k$ would be independent and can be factored in parallel.

A trivial choice for $\widetilde{V}$ is the diagonal matrix whose diagonals are $\widetilde{V}_{ii} := \max\{V_{ii}, 0\}$; if there are no equality constraints in (2.1), then $V \succeq 0$ and this choice of $\widetilde{V}$ results in $\widetilde{V}_{ii} = V_{ii}$ for all i.

Alternatively, one can compute a block-diagonal $\widetilde{V}$ by solving, for example, an optimization problem of the form

$$\min_{\widetilde{V}} \|\widetilde{V} - V\|_p \text{ subject to } \widetilde{V} \succeq 0. \tag{4.3}$$

However, this is only recommended in applications where H, F and G are constants and we need to solve a set of QPs with different h, f and/or g, as in the control problems discussed in Section 6. In these applications the optimization problem (4.3) is solved once during the off-line design phase, so the computational cost of solving (4.3) is not that important.

TABLE 5.1
Flops per IPM iteration

Method	Flops
Exact IPM with LDL^T factorization	$\frac{1}{3}(n_d+n_e)^3+3(n_d+n_e)^2+$ $2n_in_d^2+8n_in_d+n_dn_e$
Exact IPM with MINRES	$(2n_d^2+4n_dn_e)N^k_{\text{MINRES}}+$ $2n_in_d^2+8n_in_d+n_dn_e$
Inexact IPM with MINRES	$(2n_d^2+4n_dn_e)N^k_{\text{MINRES}}+$ $2n_in_d^2+8n_in_d+n_dn_e$
δ-active Exact IPM with LDL^T factorization	$\frac{1}{3}(n_a^k)^3+3(n_a^k)^2+$ $4n_d^2+2n_e^2+4n_a^kn_d+2n_tn_d+2n_a^kn_e$
δ-active Inexact IPM with MINRES	$(2(n_a^k)^2+n_a^k)N^k_{\text{MINRES}}+4n_in_d+4n_dn_e+$ $4n_d^2+2n_e^2+4n_a^kn_d+2n_tn_d+2n_a^kn_e$

5. Computational complexity analysis. In this section, we compare the computational complexity of our proposed IPM with existing IPMs in terms of the number of flops per IPM iteration. A flop is defined as one addition, subtraction, multiplication or division of two floating-point numbers. We consider two important cases in determining the computational complexity of different IPMs in solving a QP, one with dense matrices and another one with banded matrices.

5.1. Computational complexity of IPMs with dense matrices. Consider a QP of the form (2.1) in which the matrices H, F and D are dense. The complete algorithms for solving (2.1) with existing IPMs and our proposed algorithm are described in Algorithms 1 and 2, respectively. We consider three variants of Algorithm 1 in which the linear system (2.11) is solved with different solvers, namely an exact IPM with LDL^T factorization [6], an exact IPM with MINRES, and an inexact IPM with MINRES.

In the dense formulation, we prefer to solve (2.11) (or (2.12) if there are no equality constraints), which has fewer unknowns than (2.10a). The computational cost per iteration of these IPMs, along with the δ-active IPMs, is given in Table 5.1, where N^k_{MINRES} denotes the number of MINRES iterations required to solve (2.11) in the k^{th} IPM iteration. In Table 5.1 we consider the case when (2.1) has equality constraints, upper and lower bounds on x, and general inequality constraints.

In the first three IPMs we need to compute the matrix triple product according to (2.13). The first two parts in (2.13) can be computed by just picking the rows and columns of $(W_l^k)^{-1}$ and $(W_u^k)^{-1}$ according to the sets $\mathcal{I}_l$ and $\mathcal{I}_u$, respectively. The terms $\frac{1}{3}(n_d+n_e)^3$ and $2n_in_d^2$ in the cost of the exact IPM with LDL^T factorization are due to the LDL^T factorization and the triple product $D^T(W_d^k)^{-1}D$, respectively. If (2.1) has only lower and upper bounds on x, the term $2n_in_d^2$ disappears, giving a substantial reduction in the total computational cost.

Note that only the high-order terms are given in these tables. However, all terms, including the lower-order ones, are taken into account in the numerical results presented in Section 6.

5.2. Computational complexity of IPMs with banded matrices. Here we consider that the given matrices H, F and G are banded. In this case it is possible to get $\bar{H}$ and $\mathcal{A}_2^k$ into a banded form by rearranging the rows and columns. This type of QP arises in many applications, such as the optimal control problems described in

TABLE 5.2
Flops per IPM iteration with banded matrices

Method	Flops
Exact IPM with *LU* factorization	$\frac{1}{3}(b_{\mathcal{A}_2^k})^2(n_d+n_e)+5b_{\mathcal{A}_2^k}(n_d+n_e)+$ $2b_G n_i n_d+6b_G n_i+2(b_H+b_F+b_G)n_d$
Exact IPM with MINRES	$(2b_{\mathcal{A}_2^k}(n_d+n_e))N^k_{\text{MINRES}}+$ $2b_G n_i n_d+6b_G n_i+2(b_H+b_F+b_G)n_d$
Inexact IPM with MINRES	$(2b_{\mathcal{A}_2^k}(n_d+n_e))N^k_{\text{MINRES}}+$ $2b_G n_i n_d+6b_G n_i+2(b_H+b_F+b_G)n_d$
δ-active Exact IPM with *LU* factorization	$\frac{1}{3}(b_{\mathcal{A}_2^k})^2 n_a^k+5b_{\mathcal{A}_2^k}n_a^k+2b_G n_i+8b_{\bar{H}}(n_d+n_e)+$ $4b_G n_a^k+2b_G n_t+2(b_H+b_F+b_G)n_d$
δ-active Inexact IPM with MINRES	$\left(4b_{\bar{H}}(n_d+n_e)+2b_G n_a^k+n_a^k\right)N^k_{\text{MINRES}}+2b_G n_i+$ $8b_{\bar{H}}(n_d+n_e)+4b_G n_a^k+2b_G n_t+2(b_H+b_F+b_G)n_d$

Section 6.

Let us denote the bandwidths of H, F, G, $\bar{H}$ and $\mathcal{A}_2^k$ by b_H, b_F, b_G, $b_{\bar{H}}$ and $b_{\mathcal{A}_2^k}$, respectively. The computational cost of different IPMs per iteration in solving (2.1) with equality constraints, upper and lower bounds and general inequality constraints is described in Table 5.2, where we have used LU factorization instead of LDL^T factorization because the Bunch-Kaufman algorithm [6] for factoring banded symmetric indefinite matrices can destroy the banded structure of the matrix [19].

6. Application to a finite horizon optimal control problem. In this section we describe the computational issues involved in solving a finite horizon optimal control problem for a discrete-time linear system with quadratic cost and linear inequality constraints on the states and control inputs, as in predictive control applications [21]. In predictive control a suitably-defined QP is solved at each sample instant, given the current estimate of the state, to obtain the optimal sequence of inputs, after which only the first input is applied to the plant. This process is repeated at every sample instant using updated state estimates.

There are essentially two popular ways to formulate the finite horizon optimal control problem as a QP, namely the *condensed* and *non-noncondensed* formulations. In the condensed formulation, the equality constraints and state variables are eliminated by writing them as explicit linear functions of the control sequence—this results in a small but dense Hessian [21]. The number of flops in each iteration of an exact IPM using Cholesky factorization is $\mathcal{O}((l+m)m^2N^3)$, where l is the number of constrained variables, m is the number of inputs and N is the horizon length. In the non-condensed formulation, states are considered unknowns and state equations are treated as equality constraints—this results in a large but sparse Hessian [21,29]. Using a Riccati recursion scheme to solve the resulting linear system (2.11a), the number of flops in each iteration of an exact IPM can be reduced to $\mathcal{O}((n+m)^3+l(m+n)^2)N)$, where n is the number of states [26]. The non-condensed approach is therefore often preferred in situations where a long horizon length N is desirable; a bigger value of N guarantees a larger region of attraction and better closed-loop performance.

In recent years, attempts have been made to use predictive control in fast processes with a short sampling time. To reduce the computational load in solving the finite horizon optimal control problem, some new techniques have emerged [1,27,28]. In [1] a number of QPs are solved off-line, then a piecewise affine function is formed using

the solutions of the QPs. However, the number of regions describing the function may be very large and the approach is therefore usually only applicable to small-scale problems. In a second approach [28], a QP problem is solved on-line and warm-starting and early termination of the QP solver is proposed. In warm-starting, the initialization of the QP problem uses the predictions made in the previous step. This reduces the computational cost only if the new QP is similar to the previous one. Though early termination significantly reduces the computation time, it may lead to an unstable closed-loop if the equality constraints are not satisfied at termination. In [27] an iterative scheme based on fast gradient methods is described. This scheme allows one to compute a bound on the number of iterations to achieve a given level of sub-optimality, but is currently only applicable to problems with upper and lower bounds on the inputs only.

6.1. Definition of optimal control problem. Consider a discrete-time linear time-invariant system of the form [4]

$$\chi(i+1) = A\chi(i) + Bu(i), \quad \gamma(i) = C\chi(i), \tag{6.1}$$

where $\chi(i) \in \mathbb{R}^n$ is the state vector, $u(i) \in \mathbb{R}^m$ is the control input vector and $\gamma(i) \in \mathbb{R}^p$ is the output vector at the i^{th} time instant, the matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$. Let $\bar{\chi} = \chi(0) \in \mathbb{R}^n$ be the measurement or estimate of the state at the current time instant. The objective is to find, over a finite horizon of length N, a sequence of optimal control inputs $u(0), \ldots, u(N-1)$ subject to the equality constraints (6.1) and the inequality constraints

$$J_i\chi(i) + E_iu(i) \leq d_i, \quad i = 0, \ldots, N-1, \tag{6.2a}$$

$$J_N\chi(N) \leq d_N, \tag{6.2b}$$

while minimizing the quadratic cost function

$$\chi(N)^T P\chi(N) + \sum_{i=0}^{N-1} \Big(\chi(i)^T Q\chi(i) + u(i)^T Ru(i) + 2\chi(i)^T Mu(i)\Big) \tag{6.3}$$

with $R \succ 0$, $Q \succeq 0$, $Q - MR^{-1}M^T \succeq 0$, $P \succeq 0$, J_i, $J_N \in \mathbb{R}^{l\times n}$, $E_i \in \mathbb{R}^{l\times m}$, Q, $P \in \mathbb{R}^{n\times n}$, $R \in \mathbb{R}^{m\times m}$ and $M \in \mathbb{R}^{n\times m}$.

Following the non-condensed approach [26, 29], if we define the vector of decision variables as

$$x := \begin{bmatrix}\chi(0)^T & u(0)^T & \chi(1)^T & u(1)^T & \cdots & u(N-1)^T & \chi(N)^T\end{bmatrix}^T, \tag{6.4}$$

then the optimal control problem can be converted to a sparse, convex QP of the form

$$x^* := \arg\min_x \frac{1}{2}x^T Hx \quad \text{subject to} \quad Fx = f(\bar{\chi}), \quad Gx \leq g, \tag{6.5}$$

where $x \in \mathbb{R}^{n_d}$, $H \in \mathbb{R}^{n_d\times n_d}$, $F \in \mathbb{R}^{n_e\times n_d}$, $G \in \mathbb{R}^{n_i\times n_d}$ with $n_d := (n+m)N + n$, $n_e := n(N+1)$ and $n_t := l(N+1)$. The matrices H, F and G and the vectors $f(\bar{\chi})$ and d are given by

$$H := \begin{bmatrix} Q & M & \cdots & 0 & 0 & 0 \\ M^T & R & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & Q & M & 0 \\ 0 & 0 & \cdots & M^T & R & 0 \\ 0 & 0 & \cdots & 0 & 0 & P \end{bmatrix},$$

$$F := \begin{bmatrix} -I & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ A & B & -I & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & 0 & \cdots & A & B & -I \end{bmatrix}, \quad f(\bar{\chi}) := \begin{bmatrix} -\bar{\chi} \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$G := \begin{bmatrix} J_0 & E_0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & J_{N-1} & E_{N-1} & 0 \\ 0 & 0 & \cdots & 0 & 0 & J_N \end{bmatrix}, \quad g := \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_N \end{bmatrix}.$$

It follows by inspection that F has full row rank and, since $R \succ 0$, one can show that $\ker(H) \cap \ker(F) = \{0\}$. Hence $\bar{H}$, defined in (3.10a), is non-singular. It also follows that the solution x^* to the QP (6.5) is unique, if it exists.

6.2. Numerical results. We present a numerical study to evaluate the performance of our proposed method. Consider a system of p equal masses connected by springs and to walls at the ends. This example is a generalization of the example taken in [28], where only 6 masses are considered. We have selected it because we are interested in determining how the number of flops will scale with the number of states, inputs and horizon length, and we can easily change these variables. The mass of each block is 1 kg and the spring constant of each spring is taken as 1 N/m. There is no damping. The continuous-time equations of motion of the spring-mass system are described as

$$\begin{aligned} \ddot{q}_1 &= -2q_1 + q_2 + v_1 \\ \ddot{q}_2 &= q_1 - 2q_2 + q_3 + v_2 \\ &\vdots \\ \ddot{q}_m &= q_{m-1} - 2q_m + q_{m+1} + v_m \\ \ddot{q}_{m+1} &= q_m - 2q_{m+1} + q_{m+2} \\ &\vdots \\ \ddot{q}_p &= q_{p-1} - 2q_p, \end{aligned}$$

where q_i denotes the position of the i^{th} mass with respect to its equilibrium position and v_i represents the control force acting on the i^{th} mass. The state-space form can be achieved by defining

$$\begin{aligned} \chi &:= [q_1^T \; q_2^T \; \dots \; q_p^T \; \dot{q}_1^T \; \dot{q}_2^T \dots \; \dot{q}_p^T]^T, \\ u &:= [v_1^T \; v_2^T \; \dots \; v_m^T]^T, \\ \gamma &:= [q_1^T \; q_2^T \; \dots \; q_p^T]^T. \end{aligned}$$

There are m actuators connected to the first m masses and we have the following inequality constraints on the inputs and outputs:

$$\begin{aligned} -0.5 \le u(i) \le 0.5, \quad i = 0, \dots, N-1, \\ -4 \le \gamma(i) \le 4, \quad i = 1, \dots, N. \end{aligned}$$

The continuous-time state-space system is transformed into an equivalent discrete-time system using a sample time of 0.5 s while keeping the inputs constant in between

TABLE 6.1
Flops per IPM iteration for the optimal control problem

Method	**Flops**
Exact IPM with Riccati Recursion [26]	$(3n^3 + 6n^2m + 3nm^2 + \frac{1}{3}m^3)N + l(n^2 + 2nm + m^2)N$
Exact IPM with *LU* factorization	$2(8n^3 + 6n^2m + 6nm^2 + m^3)N + l(n^2 + 2nm + m^2)N$
Exact IPM with MINRES	$(2(2n+m)^2N)N^k_{\text{MINRES}} + l(n^2 + 2nm + m^2)N$
Inexact IPM with MINRES	$(2(2n+m)^2N)N^k_{\text{MINRES}} + l(n^2 + 2nm + m^2)N$
δ-active Inexact IPM with MINRES	$\left(4(2n+m)^2N + 2(n+m)n^k_a\right) N^k_{\text{MINRES}}$

sample instants. The objective is to regulate the displacements with the given constraints on displacements and control inputs. The regulator tuning matrices are taken as $R := I, M := 0$ and $Q := C^TC = [I_p\ 0]^T[I_p\ 0]$. In order to ensure local stability as in [26], P is computed from the following discrete-time algebraic Riccati equation:

$$P = A^TPA + Q - (B^TPA + M)^T(R + B^TPB)^{-1}(B^TPA + M). \tag{6.6}$$

Several simulations are carried out with initial conditions

$$\bar{\chi} = 3.5[1\ 1\ 0\ \cdots\ 0]^T,\ x^0 = \mathbf{1}_{n_d},\ y^0 = \mathbf{1}_{n_e},\ z^0 = s^0 = \mathbf{1}_{n_i},$$
$$\epsilon = 10^{-3},\ \delta = 1.5,\ \zeta = 6,\ \beta = 0.87,\ \sigma = 0.2.$$

The values of δ, z^0 and s^0 are selected such that all inequality constraints are δ-active at $k = 0$.

Fig. 6.1(a) indicates that the rate of convergence of MINRES is very slow on the original linear system (2.10a), but much faster for the smaller modified system (3.9). This rate of convergence is further enhanced using a preconditioned MINRES (PMINRES) method with a diagonal preconditioner $\mathcal{P}_2^k$, computed as in (4.3) with $p = \infty$. It is also observed in some cases that MINRES fails to converge when solving (2.10a) and the solution never reaches the desired accuracy, due to the high condition number of $\mathcal{A}_1^k$. Fig. 6.1(b) indicates that the condition number of $\mathcal{A}_1^k$ increases while the condition number of the modified matrix in (3.9) remains almost constant as the iteration k of Algorithm 1 increases. Fig. 6.1(c) indicates that the number of δ-active inequality constraints decreases as k increases. Fig. 6.1(d) indicates that the normalized error in the solution of the modified system (3.8a) decreases as the IPM converges, because the tolerance η^k decreases to zero.

The number of flops per iteration of an IPM that uses the Riccati recursion method to compute the search direction [26] is given in Table 6.1, along with other IPMs as described in Section 5. It is evident that the computational complexity of an exact IPM with *LU* factorization is higher than the Riccati recursion method. The rate of convergence of MINRES on the original system (2.10a) is very slow, but much faster for the modified system (3.9), as indicated in Fig. 6.1(a). Therefore, in the numerical results that follow, we only compare the δ-active based method against the Riccati recursion method.

To see the growth of computational cost with the number of states n, simulations were carried out with a fixed number of inputs m and horizon length N. Fig. 6.2(a)

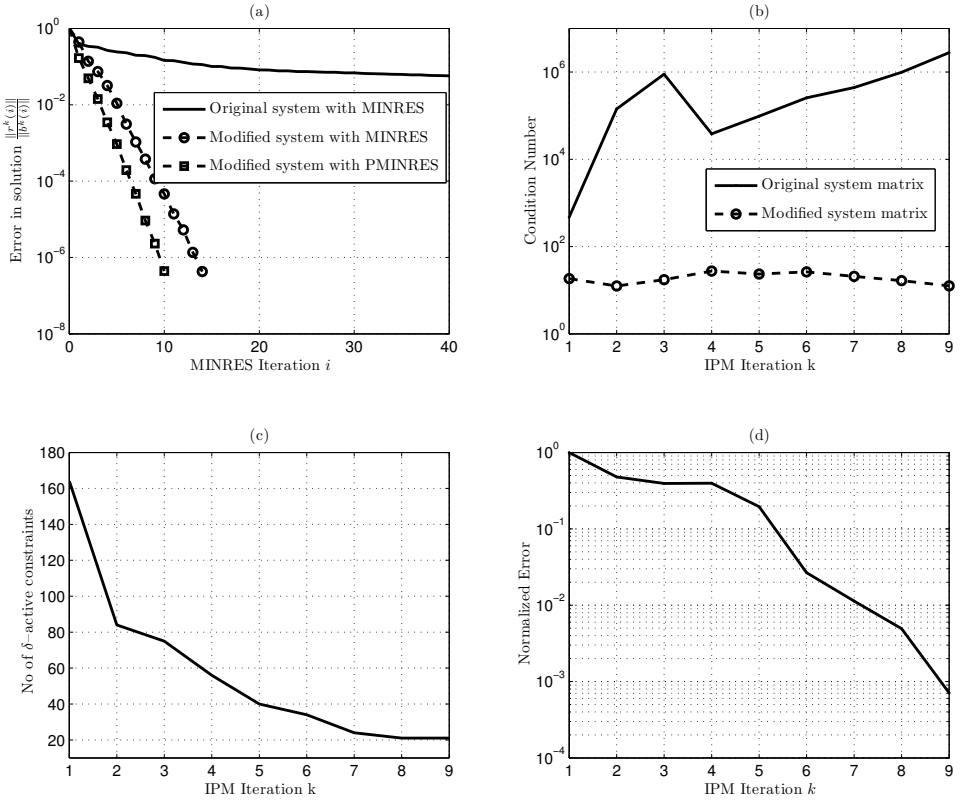

FIG. 6.1. *Results for a system with $n = 10$, $m = 2$ and $N = 20$ (a) Comparison of rate of convergence on solving the original linear system (2.10) and approximate linear system (3.8) or (3.9) with MINRES and PMINRES for $k = 3$. (b) Comparison of condition numbers of original matrix $\mathcal{A}_1^k$ of (2.10a) and modified matrix A_1^k of (3.7a). (c) Decay of number of δ-active inequality constraints with main loop iteration k. (d) Decay of normalized error $\|e^k\|_\infty/\|e^0\|_\infty$ in the solution of the modified system (3.8), where $\|e^k\|_\infty := \left\|\begin{bmatrix} e_1^k & e_2^k \end{bmatrix}\right\|_\infty$. The upper bounds on e_1^k and e_2^k are defined in (3.13).*

shows that the computational complexity of MINRES and PMINRES with the δ-active inexact IPM is less than the Riccati recursion method. In PMINRES, the preconditioner is selected as $\mathcal{P}_1^k$ if $n_a^k \geq n_d$ and $\mathcal{P}_2^k$ if $n_a^k < n_d$. The plots of $n^2/180$ and $n^3/1300$ are also given for comparison. Note that MINRES and PMINRES roughly scale with $\mathcal{O}(n^2)$. Secondly, with n and m fixed, simulations were carried out for varying N and results are plotted in Fig. 6.2(b), which indicates that MINRES and PMINRES roughly scale with $\mathcal{O}(N)$. Thirdly, with n and N fixed, simulations were carried out for varying m and results are plotted in Fig. 6.2(c), indicating that the computational complexity of MINRES and PMINRES is less than the Riccati recursion method.

7. Conclusions. In each iteration of an IPM, a system of linear equations ((2.10a), (2.11a) or (2.12)) needs to be solved to find the search direction. This system becomes increasingly ill-conditioned as the IPM iterations converge, which restricts the use of iterative methods. Handling the ill-conditioning of this linear system in the later iterations of an IPM is arguably still an open issue.

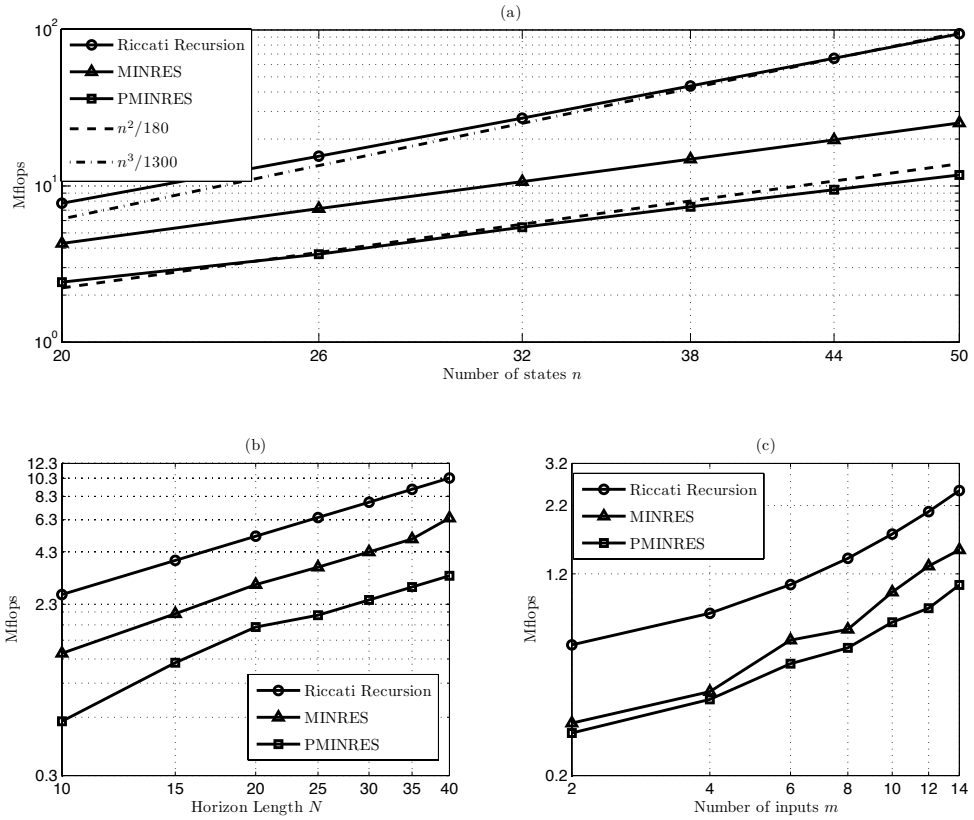

FIG. 6.2. *Mflops represents the total number of flops required to solve (6.5) by Algorithm 2 and the Riccati recursion method. (a) Growth of number of flops with number of states n, where $m = 1$ and $N = 30$. (b) Growth of number of flops with horizon length N, where $n = 20$ and $m = 1$. (c) Growth of number of flops with inputs m, where $n = 14$ and $N = 4$.*

In this paper, we have proposed solving a smaller, approximate linear system (3.9), for which analytical and numerical results suggest that the matrix of the new linear system has a more favorable distribution of eigenvalues and condition number. We provided an upper bound on the error of the approximation, which decreases as the IPM converges. The size of the linear system can be further reduced to the number of so-called δ-active constraints. Through this strategy we have transformed a large, ill-conditioned system to a smaller, well-conditioned one, facilitating the use of iterative methods.

In Section 4 we introduced a new preconditioner and upper bound for the MINRES (CG) method for the solution of a perturbed linear system with a symmetric (symmetric and positive definite) matrix. These results are quite general in the sense that they can be used for any low-rank update of a linear system, providing a good alternative to the Sherman-Morrison-Woodbury formula or the update of Cholesky factors, particularly for sparse matrices.

Results obtained from numerical simulations indicate that the computational cost of our proposed method scales well when applied to a finite horizon optimal control problem. It was found that the modified, well-conditioned linear system can be solved

using PMINRES with fewer iterations compared to solving the original ill-conditioned system.

In our proposed new IPM, the selection of δ is important because the cost of the proposed algorithm depends heavily on this parameter. Further investigations could be undertaken into developing an efficient algorithm for the automatic selection of δ at the beginning of an IPM.

For future work, it would also be interesting to compare the performance of the proposed method in terms of CPU time rather than flops using various test problems, such as the CUTEr testing environment [17].

Acknowledgments. The authors would like to thank the anonymous reviewers for their time, in-depth reviews and suggestions, which improved the text considerably.

REFERENCES

[1] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, *The explicit linear quadratic regulator for constrained systems*, Automatica, 38 (2002), pp. 3–20.
[2] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solutions of saddle point problems*, Acta Numerica, 14 (2005), pp. 1–137.
[3] L. Bergamaschi and J. G. Z. Gondzio, *Preconditioning indefinite systems in interior point methods for optimization*, Comput. Optim. Appl., 28 (2004), pp. 149–171.
[4] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1, Athena Scientific, third ed., 2005.
[5] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, UK, 2004.
[6] J. R. Bunch and L. Kaufman, *Some stable methods for calculating inertia and solving symmetric linear systems*, American Mathematical Society, 31 (1977), pp. 163–179.
[7] S. Cafieri, *On the application of iterative solvers to KKT systems in Interior Point methods for Large-Scale Quadratic Programming problems*, PhD thesis, Universitá degli Studi di Napoli Federico II, 2006.
[8] G. A. Constantinides, *Tutorial paper: Parallel architectures for model predictive control*, in Proc. European Control Conference, Budapest, Hungary, August 2009, pp. 138–143.
[9] T. A. Davis, *Direct Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2006.
[10] R. S. Dembo, S. C. Eisentat, and T. Steihaug, *Inexact Newton methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 400–408.
[11] R. S. Dembo and T. Steihaug, *Truncated Newton algorithms for large-scale unconstrained optimization*, Mathematical Programming, 26 (1983), pp. 190–212.
[12] R. R. Fletcher, *Practical Methods of Optimization*, John Wiley, Cornwall, UK, 2nd edition 1987.
[13] R. W. Freund and F. Jarre, *A QMR-based interior-point algorithm for solving linear programs*, Mathematical Programming, 76 (1996), pp. 183–210.
[14] E. M. Gertz and J. D. Griffin, *Using an iterative linear solver in an interior point method for generating support vector machines*, Computational Optimization and Applications, 47 (2010), pp. 431–453.
[15] E. M. Gertz and S. J. Wright, *Object-oriented software for quadratic programming*, ACM Transactions on Mathematical Software, 29 (2003. See also `http://www.cs.wisc.edu/swright/ooqp`), pp. 58–81.
[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, USA, third ed., 1996.
[17] N. Gould, D. Orban, and P. Toint, *CUTEr and SifDec: a constrained and unconstrained testing environment*, ACM Trans. Math. Softw., 29 (2003), pp. 373–394.
[18] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.
[19] M. T. Jones and M. L. Patrick, *Bunch-Kaufman factorization for real symmetric indefinite banded matrices*, SIAM Journal of Matrix Analysis and Applications, 14 (1993), pp. 553–559.
[20] C. Keller, N. Gould, and A. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300–1317.
[21] J. M. Maciejowski, *Predictive Control with Constraints*, Prentice Hall, UK, 2002.
[22] MOSEK ApS, *The MOSEK optimization tools manual.* `www.mosek.com`, 2012.

[23] S. G. Nash, *A survey of truncated-Newton methods*, Journal of Computational and Applied Mathematics, 124 (2000), pp. 45–59.
[24] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, 1999.
[25] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numerical Analysis, 12 (1975), pp. 617–629.
[26] C. V. Rao, S. J. Wright, and J. B. Rawlings, *Application of interior point methods to model predictive control*, Journal of Optimization Theory and Applications, 99 (1998), pp. 723–757.
[27] S. Richter, C. N. Jones, and M. Morari, *Real-time input-constrained MPC using fast gradient methods*, in Proc. IEEE Conference on Decision and Control, Shanghai, China, December 2009.
[28] Y. Wang and S. Boyd, *Fast model predictive control using online optimization*, IEEE Transactions on Control Systems Technology, 18 (2010), pp. 267–278.
[29] S. J. Wright, *Interior point methods for optimal control of discrete time systems*, Journal of Optimization Theory and Applications, 77 (1993), pp. 161–187.
[30] ———, *Primal-Dual Interior Point Methods*, SIAM, Philadelphia, 1997.