A New Preconditioning Approach for an Interior Point–Proximal Method of Multipliers for Linear and Convex Quadratic Programming

Luca Bergamaschi, Jacek Gondzio, Ángeles Martínez,
John W. Pearson, Spyridon Pougkakiotis

December 20, 2019

Abstract

In this paper, we address the efficient numerical solution of linear and quadratic programming problems, often of large scale. With this aim, we devise an infeasible interior point method, blended with the proximal method of multipliers, which in turn results in a primal-dual regularized interior point method. Application of this method gives rise to a sequence of increasingly ill-conditioned linear systems which cannot always be solved by factorization methods, due to memory and CPU time restrictions. We propose a novel preconditioning strategy which is based on a suitable sparsification of the normal equations matrix in the linear case, and also constitutes the foundation of a block-diagonal preconditioner to accelerate MINRES for linear systems arising from the solution of general quadratic programming problems. Numerical results for a range of test problems demonstrate the robustness of the proposed preconditioning strategy, together with its ability to solve linear systems of very large dimension.

1 Introduction

In this paper, we consider linear and quadratic programming (LP and QP) problems of the following form:

$$\min_x \ \big(c^T x + \frac{1}{2}x^T Q x\big), \quad \text{s.t. } Ax = b, \ x \geq 0, \tag{1.1}$$

where $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$. For quadratic programming problems we have that $Q \succeq 0 \in \mathbb{R}^{n\times n}$, while for linear programming $Q = 0$. The problem (1.1) is often referred to as the primal form of the quadratic programming problem; the dual form of the problem is given by

$$\max_{x,y,z} \ \big(b^T y - \frac{1}{2}x^T Q x\big), \quad \text{s.t. } -Qx + A^T y + z = c, \ z \geq 0, \tag{1.2}$$

where $z \in \mathbb{R}^n$, $y \in \mathbb{R}^m$. Problems of linear or quadratic programming form are fundamental problems in optimization, and arise in a wide range of scientific applications.

A variety of optimization methods exist for solving the problem (1.1). Two popular and successful approaches are interior point methods (IPMs) and proximal methods of multipliers (PMMs). Within an IPM, a Lagrangian is constructed involving the objective function and the equality constraints of (1.1), to which a logarithmic barrier function is then added in place of the inequality constraints. Hence, a logarithmic barrier sub-problem is solved at each iteration of the algorithm (see [12] for a survey on IPMs). The key feature of a PMM is that, at each iteration, one seeks the minimum of the problem (1.1) as stated, but one adds to the objective function a penalty term involving the norm of the difference between x and the previously computed estimate. Then, an augmented Lagrangian method is applied to approximately solve each such sub-problem (see [25, 32] for a review of proximal point methods, and [5, 13, 30, 31] for a review of augmented Lagrangian methods). In this paper we consider a blend of an infeasible IPM and a PMM, which can itself be thought of as a primal-dual regularized IPM. We refer to [28] for a derivation of this approach as well as a proof of polynomial complexity. There are substantial advantages of applying regularization within IPMs, and the reliability and fast convergence of the hybrid IP–PMM make it an attractive approach for tackling linear and quadratic programming problems.

Upon applying such a technique, the vast majority of the computational effort arises from the solution of the resulting linear systems at each IP–PMM iteration. Such a system can be in the form of an augmented system, or the reduced normal equations: we focus much of our attention on the augmented system, as unless Q has some convenient structure it is highly undesirable to form the normal equations or apply the resulting matrix within a solver. Within the linear algebra community, direct methods are popular for solving such systems due to their generalizability, however if the matrix system becomes sufficiently large the storage and/or operation costs can rapidly become excessive, depending on the computer architecture used. The application of iterative methods, for instance those based around Krylov subspace methods such as the Conjugate Gradient method (CG) [14] or MINRES [24], is an attractive alternative, but if one cannot construct suitable preconditioners which can be applied within such solvers then convergence can be prohibitively slow, and indeed it is possible that convergence is not achieved at all. The development of powerful preconditioners is therefore crucial.

A range of general preconditioners have been proposed for augmented systems arising from optimization problems, see [3, 4, 9, 10, 34] for instance. However, as is the case within the field of preconditioning in general, these are typically sensitive to changes in structure of the matrices involved, and can have substantial memory requirements. Preconditioners have also been successfully devised for specific classes of programming problems solved using similar optimization methods: applications include those arising from multicommodity network flow problems [7], stochastic programming problems [6], formulations within which the constraint matrix has primal block-angular structure [8], and PDE-constrained optimization problems [26, 27]. However, such preconditioners exploit particular structures arising from specific applications; unless there exists such a structure which hints as to the appropriate way to develop a solver, the design of bespoke preconditioners remains a challenge.

It is therefore clear that a completely robust preconditioner for linear and quadratic programming does not currently exist, as available preconditioners are either problem-sensitive (with a possibility of failure when problem parameters or structures are modified), or are tailored towards specific classes of problems. This paper therefore aims to provide a first step towards the construction of generalizable preconditioners for linear and quadratic programming problems. A particular incentive for this work is so that, when new application areas arise that require the solution of large-scale matrix systems, the preconditioning strategy proposed here could form the basis of a fast and feasible solver.

This paper is structured as follows. In Section 2 we describe the IP–PMM approach used to tackle linear and quadratic programming problems, and outline our preconditioning approach. In Section 3 we carry out spectral analysis for the resulting preconditioned matrix systems. In Section 4 we describe the implementation details of the method, and in Section 5 we present numerical results obtained using the inexact IP–PMM approach. In particular we present the results of our preconditioned iterative methods, and demonstrate that our new solvers lead to rapid and robust convergence for a wide class of problems. Finally, in Section 6 we give some concluding remarks.

Notation: For the rest of this manuscript, superscripts of a vector (or matrix, respectively) will denote the respective components of the vector, i.e. x^j (or $M^{(i,j)}$, respectively). Given a set (or two sets) of indices $\mathcal{I}$ (or $\mathcal{I}$, $\mathcal{J}$), the respective sub-vector (sub-matrix), will be denoted as $x^{\mathcal{I}}$ (or $M^{(\mathcal{I},\mathcal{J})}$). Furthermore, the j-th row (or column) of a matrix M is denoted as $M^{(j,:)}$ ($M^{(:,j)}$, respectively). Given an arbitrary square (or rectangular) matrix A, then $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ (or $\sigma_{\max}(A)$ and $\sigma_{\min}(A)$) denote the largest and smallest eigenvalues (singular values) of the matrix A, respectively. Given a square matrix M we denote as $W(M)$ its numerical range , defined as

$$W(M) = \left\{ z \in \mathbb{R} \text{ such that } z = \frac{x^T M x}{x^T x}, \text{for some } x \in \mathbb{R}^n \right\}.$$

Given a square matrix $Q \in \mathbb{R}^{n \times n}$, $\mathrm{diag}(Q)$ denotes the diagonal matrix satisfying: $(\mathrm{diag}(Q))^{(i,i)} =$

$Q^{(i,i)}$, for all $i \in \{1,\ldots,n\}$. Finally, given a vector $x \in \mathbb{R}^n$, we denote by X the diagonal matrix satisfying $X^{(i,i)} = x^i$, for all $i \in \{1,\ldots,n\}$.

2 Algorithmic Framework

In this section we derive a counterpart of the Interior Point-Proximal Method of Multipliers (IP–PMM) presented in [28] for solving the pair (1.1)–(1.2), that employs a Krylov subspace method for solving the associated linear systems. For a polynomial convergence result of the method, in the case where the linear systems are solved exactly, the reader is referred to [28]. Effectively, we merge the proximal method of multipliers with an infeasible interior point method, and present suitable general purpose preconditioners, using which, we can solve the resulting Newton system, at every iteration, by employing an appropriate Krylov subspace method.

Assume that, at some iteration k of the method, we have available an estimate η_k for the optimal Lagrange multiplier vector y^*, corresponding to the equality constraints of (1.1). Similarly, we denote by ζ_k the estimate of the primal solution x^*. Next, we define the proximal penalty function that has to be minimized at the k-th iteration of proximal method of multipliers, for solving (1.1), given the estimates $\eta_k,\ \zeta_k$:

$$\mathcal{L}^{PMM}_{\delta_k,\rho_k}(x;\eta_k,\zeta_k) = c^Tx + \frac{1}{2}x^TQx - \eta_k^T(Ax-b) + \frac{1}{2\delta_k}\|Ax-b\|^2 + \frac{\rho_k}{2}\|x-\zeta_k\|^2,$$

with $\delta_k > 0,\ \rho_k > 0$ some non-increasing penalty parameters. In order to solve the PMM sub-problem, we will apply one (or a few) iterations of an infeasible IPM. To do that, we alter the previous penalty function, by including logarithmic barriers, that is:

$$\mathcal{L}^{IP-PMM}_{\delta_k,\rho_k}(x;\eta_k,\zeta_k) = \mathcal{L}^{PMM}_{\delta_k,\rho_k,\eta_k,\zeta_k}(x;\eta_k,\zeta_k) - \mu_k\sum_{j=1}^{n}\ln x^j, \tag{2.1}$$

where $\mu_k > 0$ is the barrier parameter. In order to form the optimality conditions of this sub-problem, we equate the gradient of $\mathcal{L}^{IP-PMM}_{\delta_k,\rho_k}(\,\cdot\,;\eta_k,\zeta_k)$ to the zero vector, i.e.:

$$c + Qx - A^T\eta_k + \frac{1}{\delta_k}A^T(Ax-b) + \rho_k(x-\zeta_k) - \mu_kX^{-1}e_n = 0,$$

where e_n is a vector of ones of size n, and X is a diagonal matrix containing the entries of x. We define the variables: $y = \eta_k - \frac{1}{\delta_k}(Ax-b)$ and $z = \mu_kX^{-1}e_n$, to obtain the following (equivalent) system of equations:

$$\begin{bmatrix} c + Qx - A^Ty - z + \rho_k(x-\zeta_k) \\ Ax + \delta_k(y-\eta_k) - b \\ Xz - \mu_ke_n \end{bmatrix} = \begin{bmatrix} 0\\0\\0 \end{bmatrix}. \tag{2.2}$$

To solve the previous mildly nonlinear system of equations, at every iteration k, we employ Newton's method and alter its right-hand side, using a centering parameter $\sigma_k \in (0,1)$. In other words, at every iteration of IP–PMM we have available an iteration triple (x_k,y_k,z_k) and we want to solve the following system of equations:

$$\begin{bmatrix} -(Q+\rho_kI_n) & A^T & I_n \\ A & \delta_kI_m & 0 \\ Z_k & 0 & X_k \end{bmatrix}\begin{bmatrix} \Delta x_k \\ \Delta y_k \\ \Delta z_k \end{bmatrix} = \begin{bmatrix} c + Qx_k - A^Ty_k + \sigma_k\rho_k(x_k-\zeta_k) - z_k \\ b - Ax_k - \sigma_k\delta_k(y_k-\eta_k) \\ \sigma_k\mu_ke_n - X_kz_k \end{bmatrix} = \begin{bmatrix} r_{d_k} \\ r_{p_k} \\ r_{\mu_k} \end{bmatrix}. \tag{2.3}$$

We proceed by eliminating variables Δz. In particular, we have that:

$$\Delta z_k = X_k^{-1}(r_{\mu_k} - Z_k\Delta x_k),$$

where Z_k is a diagonal matrix containing the entries of z_k. Then, the augmented system that has to be solved at every iteration of IP–PMM reads as follows:

$$\begin{bmatrix} -(Q + \Theta_k^{-1} + \rho_k I_n) & A^T \\ A & \delta_k I_m \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix} = \begin{bmatrix} r_{d_k} + z_k - \sigma_k \mu_k X_k^{-1} e_n \\ r_{p_k} \end{bmatrix}, \tag{2.4}$$

where $\Theta_k = X_k Z_k^{-1}$. An important feature of the matrix Θ_k is that, as the method approaches an optimal solution, the positive diagonal matrix has some entries that (numerically) approach infinity, while others approach zero.

As we argue in the spectral analysis, in the case where $Q = 0$, or Q is diagonal, it is often beneficial to form the normal equations and approximately solve them using preconditioned CG. Otherwise, we solve system (2.4) using preconditioned MINRES. The normal equations read as follows:

$$M_{NE,k} \Delta y_k = \xi_k, \qquad M_{NE,k} = \big(A(\Theta_k^{-1} + Q + \rho_k I_n)^{-1} A^T + \delta_k I_m\big), \tag{2.5}$$

where

$$\xi_k = r_{p_k} + A(Q + \Theta_k^{-1} + \rho_k I_n)^{-1}(r_{d_k} + z_k - \sigma_k \mu_k X_k^{-1} e_n).$$

In order to employ preconditioned MINRES or CG to solve (2.4) or (2.5) respectively, we must find an approximation for the coefficient matrix in (2.5). To do so, we employ a symmetric and positive definite block-diagonal preconditioner for the saddle-point system (2.4), involving approximations for the negative of the (1,1) block, as well as the Schur complement M_{NE}. See [16, 21, 35] for motivation of such saddle-point preconditioners. In light of this, we approximate Q in the (1,1) block by its diagonal, i.e. $\tilde{Q} = \text{diag}(Q)$.

Then, we define the diagonal matrix E_k with entries

$$E_k^{(i,i)} = \begin{cases} 0 & \text{if } \big((\Theta_k^{(i,i)})^{-1} + \tilde{Q}^{(i,i)} + \rho_k\big)^{-1} < C_{E,k} \min\{\mu_k, 1\}, \\ \big((\Theta_k^{(i,i)})^{-1} + \tilde{Q}^{(i,i)} + \rho_k\big)^{-1} & \text{otherwise}, \end{cases} \tag{2.6}$$

where $i \in \{1, \ldots, n\}$, $C_{E,k}$ is a constant, and we construct the normal equations approximation $P_{NE,k} = L_M L_M^T$, by computing the (exact) Cholesky factorization of

$$P_{NE,k} = A E_k A^T + \delta_k I_m. \tag{2.7}$$

The matrix $P_{NE,k}$ acts as a preconditioner for CG applied to the normal equations. In order to construct a preconditioner for the augmented system matrix in (2.4), we employ a block-diagonal preconditioner of the form:

$$P_{AS,k} = \begin{bmatrix} \tilde{Q} + \Theta_k^{-1} + \rho_k I_n & 0 \\ 0 & P_{NE,k} \end{bmatrix}, \tag{2.8}$$

with $P_{NE,k}$ defined in (2.7). Note that MINRES requires a symmetric positive definite preconditioner and hence many other block preconditioners for (2.4) are not applicable. For example, block-triangular preconditioners, motivated by the work in [15, 21], would generally require a non-symmetric solver such as GMRES [33]. Nevertheless, block-diagonal preconditioners have been shown to be very effective in practice for problems with the block structure of (2.4) (see for example [1, 23, 35]). Furthermore, it can often be beneficial to employ CG with the preconditioner (2.7), in the case where $Q = 0$ or Q is diagonal, since the former is expected to converge faster than MINRES with (2.8). This will become clearer in the next section, where eigenvalue bounds for each of the preconditioned matrices are provided.

In view of the previous discussion, we observe that the quality of both preconditioners heavily depends on the choice of constant $C_{E,k}$, since this constant determines the quality of the approximation of the normal equations, using (2.7). In the implementation this constant is tuned dynamically, based on the quality of the preconditioner and its required memory (see Section 4). On the other hand, following the developments in [28], we tune the regularization variables $\delta_k,\ \rho_k$ based on the barrier parameter μ_k. In particular, $\delta_k,\ \rho_k$ are forced to decrease at the same rate as μ_k. The exact updates of these parameters are presented in Section 4. As we will show in the next section, this tuning choice is numerically beneficial, since if $\delta_k,\ \rho_k$ are of the same order as μ_k, then the spectrum of the preconditioned normal equations is independent of μ_k; a very desirable property for preconditioned systems arising from IPMs.

3 Spectral Analysis

3.1 Preconditioned normal equations

In this section we provide a spectral analysis of the preconditioned normal equations in the LP or separable QP case, assuming that (2.7) is used as preconditioner. Although this is a specialized setting, we may make use of the following result in our analysis of the augmented system arising from the general QP case.

Let us define this normal equations matrix $\tilde{M}_{NE,k}$, as

$$\tilde{M}_{NE,k} = A\tilde{G}_k A^T + \delta_k I_m, \quad \text{with } \tilde{G}_k = \left(\tilde{Q} + \Theta^{-1} + \rho_k I_n\right)^{-1}. \tag{3.1}$$

The following Theorem provides lower and upper bounds on the eigenvalues of $P_{NE,k}^{-1}\tilde{M}_{NE,k}$, at an arbitrary iteration k of Algorithm IP–PMM.

Theorem 3.1. *There are $m - r$ eigenvalues of $P_{NE,k}^{-1}\tilde{M}_{NE,k}$ at one, where r is the column rank of A^T, corresponding to linearly independent vectors belonging to the nullspace of A^T. The remaining eigenvalues are bounded as*

$$1 \le \lambda \le 1 + \frac{C_{E,k}\mu_k}{\delta_k}\sigma_{\max}^2(A).$$

Proof. The eigenvalues of $P_{NE,k}^{-1}\tilde{M}_{NE,k}$ must satisfy

$$A\tilde{G}_k A^T u + \delta_k u = \lambda A E_k A^T u + \lambda \delta_k u. \tag{3.2}$$

Multiplying (3.2) on the left by u^T and setting $z = A^T u$ yields

$$\lambda = \frac{z^T \tilde{G}_k z + \delta_k \|u\|^2}{z^T E_k z + \delta_k \|u\|^2} = 1 + \frac{z^T\left(\tilde{G}_k - E_k\right)z}{z^T E_k z + \delta_k \|u\|^2} = 1 + \alpha.$$

For every vector u in the nullspace of A^T we have $z = 0$ and $\lambda = 1$. The fact that both E_k and $\tilde{G}_k - E_k \succeq 0$ (from the definition of E_k) implies the lower bound. To prove the upper bound we first observe that $\lambda_{\max}(\tilde{G}_k - E_k) \le C_{E,k}\mu_k$; then

$$\alpha = \frac{z^T\left(\tilde{G}_k - E_k\right)z}{z^T E_k z + \delta_k \|u\|^2} \le \frac{z^T\left(\tilde{G}_k - E_k\right)z}{\delta_k \|u\|^2} = \frac{z^T\left(\tilde{G}_k - E_k\right)z}{\|z\|^2}\frac{1}{\delta_k}\frac{\|z\|^2}{\|u\|^2} = \frac{z^T\left(\tilde{G}_k - E_k\right)z}{\|z\|^2}\frac{1}{\delta_k}\frac{u^T A A^T u}{\|u\|^2},$$

and the thesis follows by inspecting the Rayleigh Quotients of $\tilde{G}_k - E_k$ and AA^T. □

Remark 1. *Following the discussion in the end of the previous section, we know that $\frac{\mu_k}{\delta_k} = O(1)$, since IP–PMM forces δ_k to decrease at the same rate as μ_k. Combining this with the result of Theorem 3.1 implies that the condition number of the preconditioned normal equations is asymptotically independent of μ_k.*

Remark 2. *In the LP case ($Q = 0$), or the separable QP case (Q diagonal), Theorem 3.1 characterizes the eigenvalues of the preconditioned matrix within the CG method.*

3.2 BFGS-like low-rank update of the $P_{NE,k}$ preconditioner

Given a rectangular (tall) matrix $V \in \mathbb{R}^{m\times p}$ with maximum column rank, it is possible to define a generalized block-tuned preconditioner P satisfying the property

$$P^{-1}\tilde{M}_{NE,k}V = \nu V,$$

so that the columns of V become eigenvectors of the preconditioned matrix corresponding to the eigenvalue ν. A way to construct P (or its explicit inverse) is suggested by the BFGS-based preconditioners used e.g. in [2] for accelerating Newton linear systems or analyzed in [19] for general sequences of linear systems, that is

$$P^{-1} = \nu V\Pi V^T + (I_m - V\Pi V^T\tilde{M}_{NE,k})P_{NE,k}^{-1}(I_m - \tilde{M}_{NE,k}V\Pi V^T), \quad \text{with} \quad \Pi = (V^T\tilde{M}_{NE,k}V)^{-1}.$$

Note also that if the columns of V would be chosen as e.g. the p exact rightmost eigenvectors of $P_{NE,k}^{-1}\tilde{M}_{NE,k}$ (corresponding to the p largest eigenvalues) then all the other eigenpairs,

$$(\lambda_1, z_1), \ldots, (\lambda_{m-p}, z_{m-p}),$$

of the new preconditioned matrix $P^{-1}\tilde{M}_{NE,k}$ would remain unchanged as stated in the following

Theorem 3.2. *If the columns of V are the exact rightmost eigenvectors of $P_{NE,k}^{-1}\tilde{M}_{NE,k}$ then for every $j = 1, \ldots m-p$ there holds*

$$P^{-1}\tilde{M}_{NE,k}z_j = P_{NE,k}^{-1}\tilde{M}_{NE,k}z_j = \lambda_j z_j.$$

Proof. The eigenvectors of the symmetric generalized eigenproblem $\tilde{M}_{NE,k}x = \lambda P_{NE,k}x$ form a $P_{NE,k}$-orthonormal basis and therefore $V^T P_{NE,k}z_j = V^T\tilde{M}_{NE,k}z_j = 0,\ j = 1, \ldots m-p$. Moreover, denoting with $\Lambda_p = \operatorname{diag}(\lambda_{m-p+1}, \cdots, \lambda_m)$ the diagonal matrix with the largest eigenvalues, it turns out that $\Pi = (V^T\tilde{M}_{NE,k}V)^{-1} = \Lambda_p^{-1}$. Then

$$\begin{aligned} P^{-1}\tilde{M}_{NE,k}z_j &= \nu V\Lambda_p^{-1}V^T\tilde{M}_{NE,k}z_j \\ &\quad + (I_m - V\Lambda_p^{-1}V^T\tilde{M}_{NE,k})P_{NE,k}^{-1}(\tilde{M}_{NE,k}z_j - \tilde{M}_{NE,k}V\Lambda_p^{-1}V^T\tilde{M}_{NE,k}z_j) = \\ &= (I_m - V\Lambda_p^{-1}V^T\tilde{M}_{NE,k})P_{NE,k}^{-1}\tilde{M}_{NE,k}z_j = (I_m - V\Lambda_p^{-1}V^T\tilde{M}_{NE,k})\lambda_j z_j = \lambda_j z_j. \end{aligned}$$

□

Usually columns of V are chosen as the (approximate) eigenvectors of $P_{NE,k}^{-1}\tilde{M}_{NE,k}$ corresponding to the smallest eigenvalues of this matrix. However, this choice would not produce a significant reduction in the condition number of the preconditioned matrix as the spectral analysis of Theorem 3.1 suggests a possible clustering of smallest eigenvalues around 1. We choose instead, as the columns of V, the rightmost eigenvectors of $P_{NE,k}^{-1}\tilde{M}_{NE,k}$, approximated with low accuracy by the function `eigs` of MATLAB. The ν value must be selected to satisfy $\lambda_{\min}(P_{NE,k}^{-1}\tilde{M}_{NE,k}) < \nu \ll \lambda_{\max}(P_{NE,k}^{-1}\tilde{M}_{NE,k})$. We choose $\nu = 10$, and the column size of V as $p = 10$.

Finally, by computing approximately the rightmost eigenvectors, we would expect a slight perturbation of $\lambda_1, \ldots, \lambda_{m-p}$, depending on the accuracy of this approximation.

3.3 Preconditioned augmented system

In the MINRES solution of QP instances the system matrix is

$$M_{AS,k} = \begin{bmatrix} -F_k & A^T \\ A & \delta_k I_m \end{bmatrix}, \qquad F_k = Q + \Theta_k^{-1} + \rho_k I_n$$

while the preconditioner is

$$P_{AS,k} = \begin{bmatrix} \tilde{F}_k & 0 \\ 0 & P_{NE,k} \end{bmatrix}, \qquad \tilde{F}_k = \tilde{Q} + \Theta_k^{-1} + \rho_k I_n \equiv \tilde{G}_k^{-1}.$$

The following Theorem will characterize the eigenvalues of $P_{AS,k}^{-1} M_{AS,k}$ in terms of the extremal eigenvalues of the preconditioned (1,1) block of (2.4), $\tilde{F}_k^{-1} F_k$, and of $P_{NE,k}^{-1} \tilde{M}_{NE,k}$ as described by Theorem 3.1. We will work with (SPD) similarity transformations of these matrices defined as

$$\hat{F}_k = \tilde{F}_k^{-1/2} F_k \tilde{F}_k^{-1/2}, \quad \hat{M}_{NE,k} = P_{NE,k}^{-1/2} \tilde{M}_{NE,k} P_{NE,k}^{-1/2}. \tag{3.3}$$

and set

$$\begin{array}{lllllllll} \alpha_{NE} & = & \lambda_{\min}(\hat{M}_{NE,k}), & \beta_{NE} & = & \lambda_{\max}\left(\hat{M}_{NE,k}\right), & \kappa_{NE} & = & \dfrac{\beta_{NE}}{\alpha_{NE}}, \\ \alpha_F & = & \lambda_{\min}\left(\hat{F}_k\right), & \beta_F & = & \lambda_{\max}\left(\hat{F}_k\right), & \kappa_F & = & \dfrac{\beta_F}{\alpha_F}. \end{array}$$

Hence, an arbitrary element of the numerical range of these matrices is represented as:

$$\gamma_{NE} \in W(\hat{M}_{NE,k}) = [\alpha_{NE}, \beta_{NE}], \qquad \gamma_F \in W(\hat{F}_k) = [\alpha_F, \beta_F].$$

Similarly, an arbitrary element of $W(P_{NE,k})$ is denoted by

$$\gamma_p \in [\lambda_{\min}(P_{NE,k}), \lambda_{\max}(P_{NE,k})] \subseteq \left[\delta_k, \frac{\sigma_{\max}^2(A)}{\rho_k} + \delta_k\right).$$

Observe that $\alpha_F \leq 1 \leq \beta_F$ as

$$\frac{1}{n} \sum_{i=1}^n \lambda_i \left(\tilde{F}_k^{-1} F_k\right) = \frac{1}{n} \operatorname{Tr}\left(\tilde{F}_k^{-1} F_k\right) = 1.$$

Theorem 3.3. *Let k be an arbitrary iteration of IP–PMM. Then, the eigenvalues of $P_{AS,k}^{-1} M_{AS,k}$ lie in the union of the following intervals:*

$$I_- = \left[-\beta_F - \sqrt{\beta_{NE}}, -\alpha_F\right]; \quad I_+ = \left[\frac{1}{1+\beta_F}, 1 + \sqrt{\beta_{NE} - 1}\right].$$

Proof. The eigenvalues of $P_{AS,k}^{-1} M_{AS,k}$ are the same as those of

$$P_{AS,k}^{-1/2} M_{AS,k} P_{AS,k}^{-1/2} = \begin{bmatrix} \tilde{F}_k^{-1/2} & 0 \\ 0 & P_{NE,k}^{-1/2} \end{bmatrix} \begin{bmatrix} -F_k & A^T \\ A & \delta_k I_m \end{bmatrix} \begin{bmatrix} \tilde{F}_k^{-1/2} & 0 \\ 0 & P_{NE,k}^{-1/2} \end{bmatrix} = \begin{bmatrix} -\hat{F}_k & R_k^T \\ R_k & \delta_k P_{NE,k}^{-1} \end{bmatrix},$$

where $\hat{F}_k$ is defined in (3.3) and $R_k = P_{NE,k}^{-1/2} A \tilde{F}_k^{-1/2}$.

Any eigenvalue λ of $P_{AS,k}^{-1/2} M_{AS,k} P_{AS,k}^{-1/2}$ must therefore satisfy

$$-\hat{F}_k w_1 + R_k^T w_2 = \lambda w_1 \tag{3.4}$$

$$R_k w_1 + \delta_k P_{NE,k}^{-1} w_2 = \lambda w_2. \tag{3.5}$$

First note that

$$R_k R_k^T = P_{NE,k}^{-1/2} A \tilde{F}_k^{-1} A^T P_{NE,k}^{-1/2} = P_{NE,k}^{-1/2} \left(\tilde{M}_{NE,k} - \delta_k I_m\right) P_{NE,k}^{-1/2} = \hat{M}_{NE,k} - \delta_k P_{NE,k}^{-1}. \tag{3.6}$$

The eigenvalues of $R_k R_k^T$ are therefore characterized by Theorem 3.1. If $\lambda \notin [-\beta_F, -\alpha_F]$ then $\hat{F}_k + \lambda I_n$ is symmetric positive (or negative) definite; moreover $R_k^T w_2 \neq 0$. Then from (3.4) we obtain an expression for w_1

$$w_1 = (\hat{F}_k + \lambda I_n)^{-1} R_k^T w_2,$$

which, after substituting in (3.5) yields

$$R_k(\hat{F}_k + \lambda I_n)^{-1} R_k^T w_2 + \delta_k P_{NE,k}^{-1} w_2 = \lambda w_2.$$

Premultiplying by w_2^T and dividing by $\|w_2\|^2$, we obtain the following equation where we have set $z = R_k^T w_2$.

$$\lambda = \frac{z^T(\hat{F}_k + \lambda I_n)^{-1} z}{z^T z} \frac{w_2^T R_k R_k^T w_2}{w_2^T w_2} + \delta_k \frac{w_2^T P_{NE,k}^{-1} w_2}{w_2^T w_2} = \frac{1}{\gamma_F + \lambda}\left(\gamma_{NE} - \frac{\delta_k}{\gamma_p}\right) + \frac{\delta_k}{\gamma_p}.$$

So λ must satisfy the following second-order algebraic equation

$$\lambda^2 + (\gamma_F - \omega)\,\lambda - (\omega(\gamma_F - 1) + \gamma_{NE}) = 0.$$

where we have set $\omega = \dfrac{\delta_k}{\gamma_p}$ satisfying $\omega \leq 1$ for all $k \geq 0$.
We first consider the negative eigenvalue solution of the previous algebraic equation, that is:

$$\begin{aligned}
\lambda_- &= \frac{1}{2}\left[\omega - \gamma_F - \sqrt{(\gamma_F - \omega)^2 + 4(\omega\gamma_F - \omega + \gamma_{NE})}\right] \\
&= \frac{1}{2}\left[\omega - \gamma_F - \sqrt{(\gamma_F + \omega)^2 + 4(\gamma_{NE} - \omega)}\right] \\
&\leq \frac{1}{2}\left[\omega - \gamma_F - \sqrt{(\gamma_F + \omega)^2}\right] = -\gamma_F \leq -\alpha_F,
\end{aligned}$$

In order to derive a lower bound on λ_- we work similarly. That is:

$$\begin{aligned}
\lambda_- &= \frac{1}{2}\left[\omega - \gamma_F - \sqrt{(\gamma_F + \omega)^2 + 4(\gamma_{NE} - \omega)}\right] \\
&\geq \frac{1}{2}\left[-\gamma_F - \sqrt{\gamma_F^2 + 4\gamma_{NE}}\right] \\
&\geq \frac{1}{2}\left[-\beta_F - \sqrt{\beta_F^2 + 4\beta_{NE}}\right] \geq -\beta_F - \sqrt{\beta_{NE}},
\end{aligned}$$

where we used the fact that the λ_- is an increasing function with respect to ω, and decreasing with respect to γ_{NE}. Combining all the previous yields:

$$\lambda_- \begin{cases} \geq -\beta_F - \sqrt{\beta_{NE}}, \\ \leq -\alpha_F. \end{cases}$$

Note that this interval for λ_- contains the interval $[-\beta_F, -\alpha_F]$, which we have excluded in order to carry out the analysis.

Regarding the positive eigenvalues we have that:

$$\lambda_+ = \frac{1}{2}\left[\omega - \gamma_F + \sqrt{(\gamma_F - \omega)^2 + 4(\omega\gamma_F - \omega + \gamma_{NE})}\right] = \frac{1}{2}\left[\omega - \gamma_F + \sqrt{(\gamma_F + \omega)^2 + 4(\gamma_{NE} - \omega)}\right].$$

We proceed by finding a lower bound for λ_+. To that end, we notice that λ_+ is a decreasing function with respect to the variable γ_F and increasing with respect to γ_{NE}. Hence, we have that:

$$\begin{aligned}
\lambda_+ &\geq \frac{1}{2}\left[\omega - \beta_F + \sqrt{(\beta_F + \omega)^2 + 4(\alpha_{NE} - \omega)}\right] \\
&\geq \frac{1}{2}\left[\omega - \beta_F + \sqrt{(\beta_F + \omega)^2 + 4(1 - \omega)}\right], \quad \text{since } \alpha_{NE} \geq 1, \text{ from Theorem 3.1}, \\
&\geq \frac{1}{2}\left[-\beta_F + \sqrt{\beta_F^2 + 4}\right], \quad \text{since the previous is increasing with respect to } \omega, \\
&\geq \frac{1}{1 + \beta_F}.
\end{aligned}$$

Similarly, in order to derive an upper bound for λ_+, we observe that λ_+ is an increasing function with respect to ω, decreasing with respect to γ_F and increasing with respect to γ_{NE}. Combining all the previous yields:

$$\lambda_+ \leq \frac{1}{2}\left[1 - \alpha_F + \sqrt{(\alpha_F+1)^2 + 4(\beta_{NE}-1)}\right] \leq 1 + \sqrt{\beta_{NE}-1},$$

where we used the fact that $\omega \leq 1$. Then, combining all the previous gives the desired bounds, that is:

$$\lambda_+ \begin{cases} \geq \dfrac{1}{1+\beta_F} \\ \leq 1 + \sqrt{\beta_{NE}-1}, \end{cases}$$

and completes the proof. □

Remark 3. *It is well known that a pessimistic bound on the convergence rate of MINRES can be obtained if the size of I_- and I_+ are roughly the same. In our case, as usually $\beta_F \ll \beta_{NE}$, we can assume that the length of both intervals is roughly $\sqrt{\beta_{NE}}$. As a heuristic we may therefore use* [11, Theorem 4.14]*, which predicts the reduction of the residual in the P_{AS}^{-1}-norm in the case where both intervals have exactly equal length. This then implies that*

$$\frac{\|r_k\|}{\|r_0\|} \leq 2\left(\frac{\kappa-1}{\kappa+1}\right)^{\lfloor k/2 \rfloor}$$

where

$$\begin{aligned} \kappa &\approx \frac{1+\beta_F}{\alpha_F}\left(1+\sqrt{\beta_{NE}-1}\right)(\beta_F+\sqrt{\beta_{NE}}) \leq 2\kappa_F\left(\sqrt{1+\beta_{NE}}\right)(\beta_F+\sqrt{\beta_{NE}}) \\ &\approx 2\beta_{NE}\cdot\kappa_F \leq 2\kappa_{NE}\cdot\kappa_F. \end{aligned}$$

Remark 4. *In the LP case $\tilde{F}_k = F_k$ and therefore $\kappa_F = 1$. It then turns out that $\kappa \approx 2\kappa_{NE}$. The number of MINRES iterations is then driven by $2\kappa_{NE}$ while the CG iterations depend on $\sqrt{\kappa_{NE}}$* [17]*. We highlight that different norms are used to describe the reduction in the relative residual norm for MINRES and CG.*

4 Algorithms and Implementation Details

In this section, we provide some implementation details of the method. The code was written in MATLAB and can be found here: **https://github.com/spougkakiotis/Inexact_IP–PMM** (source link). In the rest of this manuscript, when referring to CG or MINRES, we implicitly assume that the methods are preconditioned. In particular, the preconditioner given in (2.7) is employed when using CG, while the preconditioner in (2.8) is employed when using MINRES.

4.1 Free variables

The method takes as input problems in the following form:

$$\min_x \ (c^T x + \frac{1}{2}x^T Q x), \quad \text{s.t. } Ax = b, \ x^I \geq 0, \ x^F \text{ free},$$

where $I = \{1, ..., n\} \setminus F$ is the set of indices indicating the non-negative variables. In particular, if a problem instance has only free variables, no logarithmic barrier is employed and the method reduces to a standard proximal method of multipliers.

4.2 Constraint matrix scaling

In the pre-processing stage, we check if the constraint matrix is well scaled, i.e. if:

$$\Big(\max_{i\in\{1,...,m\}, j\in\{1,...,n\}} (|A^{(i,j)}|) < 10 \Big) \wedge \Big(\min_{i\in\{1,...,m\}, j\in\{1,...,n\}:\ |A^{(i,j)}|>0} (|A^{(i,j)}|) > 0.1 \Big).$$

If the previous is not satisfied, we apply geometric scaling to the rows of A, that is, we multiply each row of A by a scalar of the form:

$$d_i = \frac{1}{\sqrt{\max_{j\in\{1,...,n\}}(|A^{(i,:)}|) \cdot \min_{j\in\{1,...,n\}:\ |A^{(i,j)}|>0}(|A^{(i,:)}|)}}, \ \forall\, i \in \{1, ..., m\}.$$

4.3 Interior point-proximal method of multipliers

4.3.1 Starting point

In order to construct a reliable starting point for the method, we follow the developments in [20]. To this end, we try to solve the pair of problems (1.1)–(1.2), ignoring the non-negativity constraints.

$$\tilde{x} = A^T(AA^T)^{-1}b, \quad \tilde{y} = (AA^T)^{-1}A(c + Q\tilde{x}), \quad \tilde{z} = c - A^T\tilde{y} + Q\tilde{x}.$$

However, we regularize the matrix AA^T and employ the preconditioned CG method to solve these systems without forming the normal equations. We use the Jacobi preconditioner to accelerate CG, i.e. $P = \text{diag}(AA^T) + \delta I_m$, where $\delta = 8$ is set as the regularization parameter.

Then, in order to guarantee positivity and sufficient magnitude of x_I, z_I, we shift these components by some appropriate constants. These shift constants are the same as the ones used in the starting point developed in [20], and hence are omitted for brevity of presentation.

4.3.2 Solving the Newton system

The Newton step is computed using a predictor–corrector method. We provide the algorithmic scheme in Algorithm PC, and the reader is referred to [20] for a complete presentation of the method. We solve the systems (4.1) and (4.2), using the proposed preconditioned iterative methods (i.e. CG or MINRES). Note that in case CG is employed, we apply it on the normal equations of each respective system. Since we restrict the maximum number of Krylov iterations, we must also check whether the solution is accurate enough. If it is not, we drop the computed directions and improve our preconditioner. If this happens for 10 consecutive iterations, the algorithm is terminated.

4.3.3 Regularization parameters

The PMM parameters are initialized as follows: $\delta_0 = 8,\ \rho_0 = 8,\ \lambda_0 = y_0,\ \zeta_0 = x_0$. At the end of every iteration, we employ the algorithmic scheme given in Algorithm PEU. In order to ensure numerical stability, δ and ρ are not allowed to become smaller than a suitable positive threshold, reg_{thr}. We set $\text{reg}_{thr} = \max\big\{\frac{\text{tol}}{\max\{\|A\|_\infty^2, \|Q\|_\infty^2\}}, 10^{-13}\big\}$. This value is based on the developments in [29], where it is shown that such a constant introduces a controlled perturbation in the eigenvalues of the non-regularized linear system. If numerical instability is detected while solving the Newton system, we increase the regularization parameters $(\delta,\ \rho)$ by a factor of 2 and solve the Newton system again. If this happens while either δ or ρ have reached their minimum value, we also increase this threshold. If the threshold is increased 10 times, the method is terminated with a message indicating ill-conditioning.

Algorithm PC Predictor–Corrector Method

Compute the predictor:

$$\begin{bmatrix} -(Q+\Theta^{-1}+\rho_k I_n) & A^T \\ A & \delta_k I_m \end{bmatrix} \begin{bmatrix} \Delta_p x \\ \Delta_p y \end{bmatrix} = \begin{bmatrix} c + Qx_k - A^T y_k - \rho_k(x_k - \zeta_k) - d_1 \\ b - Ax_k - \delta_k(y_k - \eta_k) \end{bmatrix}, \tag{4.1}$$

where $d_1^I = -\mu_k (X^I)^{-1} e_{|I|}$ and $d_1^F = 0$.
Retrieve $\Delta_p z$:

$$\Delta_p z^I = d_1^I - (X^I)^{-1}(Z^I \Delta_p x^I), \quad \Delta_p z^F = 0.$$

Compute the step in the non-negativity orthant:

$$\alpha_x^{\max} = \min_{(\Delta_p x^{I(i)}<0)} \left\{1, -\frac{x^{I(i)}}{\Delta_p x^{I(i)}}\right\}, \quad \alpha_z^{\max} = \min_{(\Delta_p z^{I(i)}<0)} \left\{1, -\frac{z_k^{I(i)}}{\Delta_p z^{I(i)}}\right\},$$

for $i = 1, ..., |I|$, and set:

$$\alpha_x = \tau \alpha_x^{\max}, \quad \alpha_z = \tau \alpha_z^{\max},$$

with $\tau = 0.995$ (avoid going too close to the boundary).
Compute a centrality measure:

$$g_\alpha = (x^I + \alpha_x \Delta_p x^I)^T (z^I + \alpha_z \Delta_p z^I).$$

Set: $\mu = \left(\frac{g_\alpha}{(x_k^I)^T z_k^I}\right)^2 \frac{g_\alpha}{|I|}$
Compute the corrector:

$$\begin{bmatrix} -(Q+\Theta^{-1}+\rho_k I_n) & A^T \\ A & \delta_k I_m \end{bmatrix} \begin{bmatrix} \Delta_c x \\ \Delta_c y \end{bmatrix} = \begin{bmatrix} d_2 \\ 0 \end{bmatrix}, \tag{4.2}$$

with $d_2^I = \mu (X^I)^{-1} e^{|I|} - (X^I)^{-1} \Delta_p X^I \Delta_p z^I$ and $d_2^F = 0$.
Retrieve $\Delta_c z$:

$$\Delta_c z^I = d_2^I - (X^I)^{-1}(Z^I \Delta_c x^I), \quad \Delta_c z^F = 0.$$

$$(\Delta x, \Delta y, \Delta z) = (\Delta_p x + \Delta_c x, \Delta_p y + \Delta_c y, \Delta_p z + \Delta_c z).$$

Compute the step in the non-negativity orthant:

$$\alpha_x^{\max} = \min_{\Delta x^{I(i)}<0} \left\{1, -\frac{x^{I(i)}}{\Delta x^{I(i)}}\right\}, \quad \alpha_z^{\max} = \min_{\Delta z^{I(i)}<0} \left\{1, -\frac{z^{I(i)}}{\Delta z^{I(i)}}\right\},$$

and set:

$$\alpha_x = \tau \alpha_x^{\max}, \quad \alpha_z = \tau \alpha_z^{\max}.$$

Update:

$$(x_{k+1}, y_{k+1}, z_{k+1}) = (x_k + \alpha_x \Delta x, y_k + \alpha_z \Delta y, z_k + \alpha_z \Delta z).$$

4.3.4 Low-rank updates for the normal equations' preconditioner

At each IP–PMM iteration we check the number of non-zeros of the preconditioner used in the previous iteration. If this number exceeds some predefined constant (depending on the number of constraints m), we perform certain low-rank updates to the preconditioner, to ensure that its quality is improved, without having to use a lot of memory. In such a case, the following tasks are performed as sketched in Algorithm LRU-0. Then, at every Krylov iteration, the computation of the preconditioned residual $\hat{r} = Pr$ requires the steps outlined in Algorithm LRU-1.

Algorithm PEU Penalty and Estimate Updates

$r = \frac{|\mu_k - \mu_{k+1}|}{\mu_k}$ (rate of decrease of μ).
if $(\|Ax_{k+1} - b\| \leq 0.95 \cdot \|Ax_k - b\|)$ **then**
  $\lambda_{k+1} = y_{k+1}$.
  $\delta_{k+1} = (1 - r) \cdot \delta_k$.
else
  $\lambda_{k+1} = \eta_k$.
  $\delta_{k+1} = (1 - \frac{1}{3}r) \cdot \delta_k$.
end if
$\delta_{k+1} = \max\{\delta_{k+1}, \text{reg}_{thr}\}$, for numerical stability (ensure quasi-definiteness).
if $(\|c + Qx_{k+1} - A^T y_{k+1} - z_{k+1}\| \leq 0.95 \cdot \|c + Qx_k - A^T y_k - z_k\|)$ **then**
  $\zeta_{k+1} = x_{k+1}$.
  $\rho_{k+1} = (1 - r) \cdot \rho_k$.
else
  $\zeta_{k+1} = \zeta_k$.
  $\rho_{k+1} = (1 - \frac{1}{3}r) \cdot \rho_k$.
end if
$\rho_{k+1} = \max\{\rho_{k+1}, \text{reg}_{thr}\}$.
$k = k + 1$.

Algorithm LRU-0 Low-Rank Updates-0: Before the Krylov Solver Iteration

Compute the p rightmost (approximate) eigenvectors $v_m, \ldots v_{m-p+1}$ of $M_{NE} v = \lambda P_{NE} v$.
Set $V = \begin{bmatrix} v_m & \ldots & v_{m-p+1} \end{bmatrix}$
Compute $Z = M_{NE} V;\ T = V^T Z;\ \Pi = T^{-1}$.

Algorithm LRU-1 Low-Rank Updates-1: Computation of $\hat{r} = Pr$

$w = \Pi(V^T r)$.
$z = r - Zw$.
Solve $P_{NE} t = z$.
$u = \Pi(Z^T t)$.
$\hat{r} = V(\nu w - u) + t$.

4.3.5 Preconditioner refinement

In Section 3, we showed that the quality of both preconditioners in (2.8) and (2.7) depends heavily on the quality of the approximation of the normal equations. In other words, the quality of the preconditioner for the normal equations in (2.7) governs the convergence of both MINRES and CG. In turn, we know from Theorem 3.1, that the quality of this preconditioner depends on the choice of the constant $C_{E,k}$, at every iteration of the method. By combining the previous with the definition of E in (2.6), we expect that as $C_{E,k}$ decreases (which potentially means that there are less zero diagonal elements in E), the quality of $P_{NE,k}$ is improved. Hence, we control the quality of this preconditioner, by adjusting the value of $C_{E,k}$.

More specifically, the required quality of the preconditioner depends on the quality of the preconditioner at the previous iteration, as well as on the required memory of the previous preconditioner. In particular, if the Krylov method converged fast in the previous IP–PMM iteration (compared to the maximum allowed number of Krylov iterations), while requiring a substantial amount of memory, then the preconditioner quality is lowered (i.e. $C_{E,k+1} > C_{E,k}$). Similarly, if the Krylov method converged slowly, the preconditioner quality is increased (i.e. $C_{E,k} > C_{E,k+1}$). If the number of non-zeros of the preconditioner is more than a predefined large constant (depending on the available memory), and the preconditioner is further not good enough, we still increase the preconditioner's quality (i.e.

we decrease $C_{E,k}$), but in a very slow rate, hoping that this happens close to convergence (which is what we observe in practice, when solving large scale problems). As a consequence, allowing more iterations for the Krylov solvers results in a (usually) slower method that requires less memory. On the other hand, by sensibly restricting the maximum number of iterations of the iterative solvers, one can achieve fast convergence, at the expense of robustness (the method is slightly more prone to inaccuracy and could potentially require more memory).

4.3.6 Termination criteria

The termination criteria of the method are summarized in Algorithm TC. In particular, the method successfully terminates if the scaled 2-norm of the primal and dual infeasibility, as well as complementarity, are less than a specified tolerance. The following two conditions in Algorithm TC are employed to detect whether the problem under consideration is infeasible. For a theoretical justification of these conditions, the reader is referred to [28]. If none of the above happens, the algorithm terminates after a pre-specified number of iterations.

Algorithm TC Termination Criteria

Input: k, tol, maximum iterations

if $\left(\frac{\|c-A^Ty+Qx-z\|}{\max\{\|c\|,1\}} \leq \text{tol}\right) \wedge \left(\frac{\|b-Ax\|}{\max\{\|b\|,1\}} \leq \text{tol}\right) \wedge \left(\mu \leq \text{tol}\right)$ **then**
  Declare convergence.
end if
if $\left(\|c + Qx_k - A^Ty_k - z_k + \rho_k(x_k - \zeta_k)\| \leq \text{tol}\right) \;\wedge\; \left(\|x_k - \zeta_k\| > 10^{10}\right)$ **then**
  if (ζ_k not updated for 5 consecutive iterations) **then**
    Declare infeasibility.
  end if
end if
if $\left(\|b - Ax_k - \delta_k(y_k - \eta_k)\| \leq \text{tol}\right) \;\wedge\; \left(\|y_k - \eta_k\| > 10^{10}\right)$ **then**
  if (η_k not updated for 5 consecutive iterations) **then**
    Declare infeasibility.
  end if
end if
if ($k >$ iterations limit) **then**
  Exit (non-optimal).
end if

5 Numerical Results

At this point, we present computational results obtained by solving a set of small to large scale linear and convex quadratic problems. Throughout all of the presented experiments, we set the maximum number of IP-PMM iterations to 200. The experiments were conducted on a PC with a 2.2GHz Intel Core i7 processor (hexa-core), 16GB RAM, run under Windows 10 operating system. The MATLAB version used was R2019a. For the rest of this section, the reported number of non-zeros of a constraint matrix of an arbitrary problem does not include possible extra entries created to transform the problem to the IP–PMM format.

Firstly, we run the method on the Netlib collection, [22]. The test set consists of 96 linear programming problems. We set the desired tolerance to tol $= 10^{-4}$. In Table 1, we collect statistics from the runs of the method over some medium scale instances of the Netlib test set (see [22]). For each problem, two runs are presented; in the first one, we solve the normal equations of systems (4.1)–(4.2) using CG, while in the second one, we solve (4.1)–(4.2) using MINRES. As we argued in Section 3, the MINRES can require more than twice as many iterations as CG to deliver an equally good direction.

Hence, we set $\text{maxit}_{\text{MINRES}} = 3 \cdot \text{maxit}_{\text{CG}} = 300$ (i.e. $\text{maxit}_{\text{CG}} = 100$). It comes as no surprise that IP–PMM with MINRES is slower, however, it allows us to solve quadratic problems whose normal equations are too expensive to be formed. More specifically, IP–PMM with CG solved the whole set successfully in 141.25 seconds, requiring 2,907 IP–PMM iterations and 101,382 CG iterations. On the other hand, IP–PMM with MINRES also solved the whole set successfully, requiring 341.23 seconds, 3,012 total IP–PMM iterations and 297,041 MINRES iterations.

Table 1: Medium Scale Linear Programming Problems

Name	**nnz(A)**	**IP–PMM: CG**			**IP–PMM: MINRES**		
		Time (s)	IP-Iter.	CG-Iter.	Time (s)	IP-Iter.	MR-Iter.
80BAU3B	29,063	3.15	48	1,886	10.74	47	4,883
D2Q06C	35,674	2.16	42	1,562	7.48	46	5,080
D6CUBE	43,888	0.97	30	933	3.26	30	3,279
DFL001	41,873	10.18	54	2,105	29.07	54	6,292
FIT2D	138,018	3.16	28	836	10.62	28	2,558
FIT2P	60,784	40.78	31	924	65.15	31	2,978
PILOT87	73,804	7.29	40	1,260	18.36	42	3,543
QAP12	44,244	4.38	14	495	8.62	14	1,465
QAP15	110,700	22.83	18	575	47.45	18	1,808

While we previously presented the runs of IP–PMM using MINRES over the Netlib collection, we did so only to compare the two variants. In particular, for the rest of this section we employ the convention that IP–PMM uses CG whenever $Q = 0$ or Q is diagonal, and MINRES whenever this is not the case. Next, we present the runs of the method over the Maros–Mészáros test set ([18]), which is comprised of 127 convex quadratic programming problems. In Table 2, we collect statistics from the runs of the method over some medium and large scale instances of the collection.

Table 2: Medium and Large Scale Quadratic Programming Problems

Name	**nnz(A)**	**nnz(Q)**	**IP–PMM**		
			Time (s)	IP-Iter.	Krylov-Iter.
AUG2DCQP	20,200	80,400	4.46	41	1,188
CONT-100	49,005	10,197	3.95	23	68
CONT-101	49,599	2,700	8.83	85	282
CONT-200	198,005	40,397	39.84	109	422
CONT-300	448,799	23,100	134,76	126	405
CVXQP1_L	14,998	69,968	54.77	111	12,565
CVXQP3_L	22,497	69,968	80.18	122	14,343
LISWET1	30,000	10,002	3.55	41	1,249
POWELL20	20,000	10,000	2.71	31	937
QSHIP12L	16,170	122,433	2.99	26	3,312

In Table 3 we collect the statistics of the runs of the method over the entire Netlib and Maros-Mészáros test sets. In particular, we solve each set with increasing accuracy and report the overall success rate of the method, the total time, as well as the total IP–PMM and Krylov iterations.
All previous experiments demonstrate that IP–PMM with the proposed preconditioning strategy inherits the reliability of IP–PMM with factorization [28], while allowing one to control the memory and processing requirements of the method (which is not the case when employing a factorization to solve the resulting Newton systems).

Most of the previous experiments were conducted on small to medium scale linear and convex

Table 3: Robustness of Inexact IP–PMM

Collection	Tol	Solved (%)	IP–PMM		
			Time (s)	IP-Iter.	Krylov-Iter.
Netlib	10^{-4}	100 %	141.25	2,907	101,482
Netlib	10^{-6}	100 %	183.31	3,083	107,911
Netlib	10^{-8}	96.87 %	337.21	3,670	119,465
Maros–Mészáros	10^{-4}	99.21 %	422.75	3,429	247,724
Maros–Mészáros	10^{-6}	97.64 %	545.26	4,856	291,286
Maros–Mészáros	10^{-8}	92.91 %	637.35	5,469	321,636

quadratic programming problems. In Table 4 we provide the statistics of the runs of the method over a small set of large scale problems. The tolerance used in these experiments was 10^{-4}.

Table 4: Large-Scale Linear Programming Problems

Name	nnz(A)	IP–PMM: CG		
		Time (s)	IP-Iter.	CG-Iter.
CONT1-l	$7,031,999$	$*$[1]	$*$	$*$
FOME13	$285,056$	72.59	54	2,098
FOME21	$465,294$	415.51	96	4,268
LP-CRE-B	$260,785$	14.25	51	2,177
LP-CRE-D	$246,614$	16.04	58	2,516
LP-KEN-18	$358,171$	128.78	42	1,759
LP-OSA-30	$604,488$	20.88	67	2,409
LP-OSA-60	$1,408,073$	56.65	65	2,403
LP-NUG-20	$304,800$	132.41	17	785
LP-NUG-30	$1,567,800$	2,873.67	22	1,141
LP-PDS-30	$340,635$	363.89	81	3,362
LP-PDS-100	$1,096,002$	3,709.93	100	6,094
LP-STOCFOR3	$43,888$	8.96	60	1,777
NEOS	$1,526,794$	† [2]	†	†
NUG08-3rd	$148,416$	80.72	17	682
RAIL2586	$8,011,362$	294.12	51	1,691
RAIL4284	$11,284,032$	391.93	46	1,567
WATSON-1	$1,055,093$	181.63	73	2,588
WATSON-2	$1,846,391$	612.68	140	5,637

We notice that the proposed version of IP–PMM is able to solve larger problems, as compared to IP–PMM using factorization (see [28], and notice that the experiments there were conducted on the same PC, using the same version of MATLAB). Moreover, for large problem instances, the use of Krylov solvers is also beneficial with respect to processing time.

6 Concluding Remarks

In this paper, we have considered a combination of the interior point method and the proximal method of multipliers to efficiently solve linear and quadratic programming problems of large size.

[1] $*$ indicates that the solver was stopped due to excessive run time.
[2] † indicates that the solver ran out of memory.

The combined method, in short IP–PMM, produces a sequence of linear systems whose conditioning progressively deteriorates as the iteration proceeds. One main contribution of this paper is the development and analysis of a novel preconditioning technique for both the normal equation system arising in LP and separable QP problems, and the augmented system for general QP instances. The preconditioning strategy consists of the construction of symmetric positive definite, block-diagonal preconditioners for the augmented system or a suitable approximation of the normal equations coefficient matrix, by undertaking sparsification of the (1,1) block with the aim of controlling the memory requirements and computational cost of the method. We have carried out a detailed spectral analysis of the resulting preconditioned matrix systems. In particular, we have shown that the spectrum of the preconditioned normal equations is independent of the logarithmic barrier parameter in the LP and separable QP cases, which is a highly desirable property for preconditioned systems arising from IPMs. We have then made use of this result to obtain a spectral analysis of preconditioned matrix systems arising from more general QP problems.

We have reported computational results obtained by solving a set of small to large linear and convex quadratic problems from the Netlib and Maros–Mészáros collections, and also large-scale linear programming problems. The experiments demonstrate that the new solver, in conjunction with the proposed preconditioned iterative methods, leads to rapid and robust convergence for a wide class of problems. We hope that this work provides a first step towards the construction of generalizable preconditioners for linear and quadratic programming problems.

Acknowledgements

This work was partially supported by the Project granted by the CARIPARO foundation *Matrix-Free Preconditioners for Large-Scale Convex Constrained Optimization Problems (PRECOOP)*. L. Bergamaschi and A. Martínez have also been supported by the INdAM Research Group GNCS (Year 2019) while J. Gondzio and S. Pougkakiotis have also been supported by the Google project *Fast interior point method for linear programming problems.*

References

[1] M. Benzi, G. H. Golub, and J. Liesen. Numerical solutions of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[2] L. Bergamaschi, R. Bru, and A. Martínez. Low-rank update of preconditioners for the inexact Newton method with SPD Jacobian. *Mathematical and Computer Modelling*, 54(7–8):1863–1873, 2011.

[3] L. Bergamaschi, J. Gondzio, M. Venturin, and G. Zilli. Inexact constraint preconditioners for linear systems arising in interior point methods. *Computational Optimization and Applications*, 36(2–3):137–147, 2007.

[4] L. Bergamaschi, J. Gondzio, and G. Zilli. Preconditioning indefinite systems in interior point methods for optimization. *Computational Optimization and Applications*, 28(2):149–171, 2004.

[5] P. D. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods.* Athena Scientific, 1996.

[6] Y. Cao, C. D. Laird, and V. M. Zavala. Clustering-based preconditioning for stochastic programs. *Computational Optimization and Applications*, 64(2):379–406, 2016.

[7] J. Castro. A specialized interior-point algorithm for multicommodity network flows. *SIAM Journal on Optimization*, 10(3):852–877, 2000.

[8] J. Castro and J. Cuesta. Quadratic regularizations in an interior-point method for primal block-angular problems. *Mathematical Programming*, 130(2):415–455, 2011.

[9] J.-S. Chai and K.-C. Toh. Preconditioning and iterative solution of symmetric indefinite linear systems arising from interior point methods for linear programming. *Computational Optimization and Applications*, 36(2–3):221–247, 2007.

[10] C. Durazzi and V. Ruggiero. Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems. *Numerical Linear Algebra with Applications*, 10(8):673–688, 2003.

[11] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*. Numerical Mathematics and Scientific Computation. Oxford University Press, 2nd edition, 2014.

[12] J. Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2013.

[13] M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4(5):303–320, 1969.

[14] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49(6):409–436, 1952.

[15] I. C. F. Ipsen. A note on preconditioning non-symmetric matrices. *SIAM Journal on Scientific Computing*, 23(3):1050–1051, 2001.

[16] Y. A. Kuznetsov. Efficient iterative solvers for elliptic finite element problems on nonmatching grids. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 10(3):187–211, 1995.

[17] J. Liesen and P. Tichý. Convergence analysis of Krylov subspace methods. *GAMM-Mitteilungen*, 27(2):153–173, 2005.

[18] I. Maros and C. Mészáros. A repository of convex quadratic programming problems. *Optimization Methods and and Software*, 11(1–4):671–681, 1999.

[19] A. Martínez. Tuned preconditioners for the eigensolution of large SPD matrices arising in engineering problems. *Numerical Linear Algebra with Applications*, 23(3):427–443, 2016.

[20] S. Mehrotra. On the implementation of a primal-dual interior-point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[21] M. F. Murphy, G. H. Golub, and A. J. Wathen. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing*, 21(6):1969–1972, 2000.

[22] Netlib. http://netlib.org/lp, 2011.

[23] Y. Notay. A new analysis of block preconditioners for saddle point problems. *SIAM Journal on Matrix Analysis and Applications*, 35(1):143–173, 2014.

[24] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[25] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 3(1):123–231, 2014.

[26] J. W. Pearson and J. Gondzio. Fast interior point solution of quadratic programming problems arising from PDE-constrained optimization. *Numerische Mathematik*, 137(4):959–999, 2017.

[27] J. W. Pearson, M. Porcelli, and M. Stoll. Interior point methods and preconditioning for PDE-constrained optimization problems involving sparsity terms. *Numerical Linear Algebra with Applications*, to appear.

[28] S. Pougkakiotis and J. Gondzio. An interior point–proximal method of multipliers for convex quadratic programming. *arXiv:1904.10369*, 2019.

[29] S. Pougkakiotis and J. Gondzio. Dynamic non-diagonal regularization in interior point methods for linear and convex quadratic programming. *Journal of Optimization Theory and Applications*, 181(3):905–945, 2019.

[30] M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*, pages 283–298. Academic Press, New York, NY, 1969.

[31] R. T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1(2):97–116, 1976.

[32] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[33] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[34] O. Schenk, A. Wächter, and M. Weiser. Inertia-revealing preconditioning for large-scale nonconvex constrained optimization. *Computational Optimization and Applications*, 31(2):939–960, 2008.

[35] D. Silvester and A. Wathen. Fast iterative solution of stabilized Stokes systems, Part II: Using general block preconditioners. *SIAM Journal on Numerical Analysis*, 31(5):1352–1367, 1994.