Tendências em Matemática Aplicada e Computacional, **20**, N. 2 (2019), 359-379

www.scielo.br/tema
doi: 10.5540/tema.2019.020.02.0359

# A New Hybrid Preconditioner for the Interior Point Method

M.R. HEREDIA[1*], C.O. CASTRO[1] and A.R.L. OLIVEIRA[2]

Received on December 19, 2017 / Accepted on March 2, 2019

**ABSTRACT.** This study aims to improve the computation of the search direction in the primal-dual Interior Point Method through preconditioned iterative methods. It is about a hybrid approach that combines the Controlled Cholesky Factorization preconditioner and the Splitting preconditioner. This approach has shown good results, however, in these preconditioners there are factors that reduce their efficiency, such as faults on the diagonal when performing the Cholesky factorization, as well as a demand for excessive memory, among others. Thus, some modifications are proposed in these preconditioners, as well as a new phase change, in order to improve the performance of the hybrid preconditioner. In the Controlled Cholesky Factorization, the parameters that control the fill-in and the correction of the faults which occur on the diagonal are modified. It considers the relationship between the components from Controlled Cholesky Factorization obtained before and after the fault on the diagonal. In the Splitting preconditioner, in turn, a sparse base is constructed through an appropriate ordering of the columns from constrained matrix optimization problem. In addition, a theoretical result is presented, which shows that, with the proposed ordering, the condition number of the preconditioned Normal Equation matrix with the Splitting preconditioner is uniformly limited by an amount that depends only on the original data of the problem and not on the iteration of the Interior Point Method. Numerical experiments with large scale problems, corroborate the robustness and computational efficiency from this approach.

**Keywords:** Interior Point Method, Controlled Cholesky Factorization, Splitting preconditioner.

## 1 INTRODUCTION

Among the Interior Point Methods (IPM) found in the literature today, the primal-dual method of infeasible points using the Mehrotra's predictor-corrector technique results to be the most computationally efficient, see [8, 9, 21]. However, the greatest computational effort in all IPM is the computation of the search direction because it results from linear systems that become ill

*Corresponding author: Manolo Rodriguez Heredia – E-mail: manolorh@unifesspa.edu.br – https://orcid.org/0000-0003-0412-4628

[1]Instituto de Engenharia do Araguaia – Universidade Federal do Sul e Sudeste do Pará (IEA – Unifesspa) E-mails: manolorh@unifesspa.edu.br, ceciliaoc@unifesspa.edu.br

[2]Instituto de Matemática, Estatística e Computação Científica – Universidade Estadual de Campinas (IMECC – Unicamp) E-mail: aurelio@ime.unicamp.br

conditioned when IPM is near to achieve the optimal solution. Additionally, this computation may require an excessive memory usage. In large scale and sparse problems, the preconditioning technique and the use of iterative methods are recommended to overcome these difficulties, see [1, 3].

The search direction can be computed by solving both as the Augmented System (AS) that has an indefinite matrix, as the Normal Equations System (NES) that has a positive definite matrix. In this paper the NES is solved using a hybrid preconditioning approach applied to the Conjugate Gradient Method (CGM). In the early iterations of the IPM is used the Controlled Cholesky Factorization preconditioner (CCF), see [4], with the proposed modifications that will be presented in Section 4.1; the objective from the contributions of this paper is to accelerate the construction of CCF preconditioner by reducing the restarts when exist diagonal faults. It was shown in [10] that the condition number of the NES matrix is the order $O(\mu^{-2})$, where $\mu$ denotes the complementarity gap of the Linear Programming (LP) problem, it means, it is inevitable that the performance of the CCF preconditioner get deteriorated when IPM is near to achieve an optimal solution since it is a generic preconditioner. Proposed by [17], the Splitting Preconditioner (SP), in turn, it was exclusively made to overcome the problem of ill conditioning of linear systems from the last IPM iterations.

The CCF preconditioner is obtained by performing an Incomplete Cholesky Factorization (ICF), its fill-in in [4] allows the preconditioner to vary from a diagonal matrix to another with more nonzero entries than the classical ICF matrix. It is known that any ICF is susceptible to faults on diagonal, however, if a symmetric matrix $V$ is positive definite, it exists a constant $\alpha > 0$ such as an ICF of the matrix $V + \alpha\,\mathrm{diag}(V)$ exists, see [15]. Techniques of diagonal modification in the ICF can be found in [11, 12, 13].

In the original CCF construction proposed in [4], the faults that occur during the factoring are corrected with an exponential increase and the computation of the elements from the preconditioner is restarted. On this study, algebraic and geometric tools are used to obtain relationships among the elements that caused the failure and the new components of the matrix obtained with the increment. In addition, it was observed that the parameter that controls the fill-in of the CCF preconditioner is related to the increase of the diagonal. Using these relations, it is proposed a modification of these parameters in order to reduce the number of factoring restarts required for its construction.

The new hybrid preconditioner is compared with the version currently used in [20]. The computational tests show that the new proposal is more efficient and robust.

We present a criterion that evaluates the performance of the CCF preconditioner and it will indicate the moment of the preconditioner exchange that starts the second phase of hybrid preconditioning using the SP. The SP performance depends on a non singular submatrix $B$ of the constraints matrix $A$, the choice of columns of $B$ is done using ordering in the columns from matrix $A$. The authors from SP and, later, their collaborators [20], developed an efficient heuristic. However, there are problems where the approach fails or demands an excessive of computational

time. Thus, in respect to SP, the objective of this paper is to study the condition number of the preconditioned NES by the SP and, from this study, to order the columns of the matrix of restrictions from LP problem in order to construct a sparse base that provides a number condition limited by an amount that is independent of the IPM iteration.

## 2 SEARCH DIRECTIONS IN THE INTERIOR POINT METHOD

Consider the linear programming problem

$$(\mathrm{P})\begin{cases}\min & c^T x\\ \text{s.t.} & Ax=b;\\ & x+s=u;\\ & x,s\geq 0,\end{cases}\quad\text{and}\quad(\mathrm{D})\begin{cases}\max & b^T y-u^T w\\ \text{s.t.} & A^T y-w+z=c;\\ & w,z\geq 0;\\ & y\in\mathbb{R}^m,\end{cases}$$

where $x,s,w\in\mathbb{R}^n$ and $A\in\mathbb{R}^{m\times n}$. We assume that A has full row rank throughout this paper. The search direction in the infeasible Interior Point Method (IPM) is obtained by applying Newton's method to the optimality conditions of the problem

$$(\mathrm{P}')\begin{cases}\min & c^T x-\mu\sum_{i=1}^n\log x_i-\mu\sum_{i=1}^n\log s_i\\ \text{s.t.} & Ax=b;\\ & x+s=u;\\ & x,s>0,\end{cases}$$

where the problem $(\mathrm{P}')$ results from applying the logarithmic barrier penalty on the non-negativity constraints of the primal problem $(P)$. Since $(\mathrm{P}')$ is a convex problem, the KKT conditions are sufficient and necessary to find the optimal solution. Consider its lagrangian $\ell$ and the partial derivatives,

$$\ell(x,s,y,w)=c^T x-\mu\sum_{i=1}^n\log x_i-\mu\sum_{i=1}^n\log s_i+y^T(b-Ax)+w^T(u-x-s),$$

$$\nabla_x\ell=c-\mu X^{-1}e-A^T y-w,\qquad \nabla_s\ell=-\mu S^{-1}e-w,$$
$$\nabla_y\ell=b-Ax\qquad\text{and}\qquad \nabla_w\ell=u-x-s,$$

where $e^T=(1,\ldots,1)\in\mathbb{R}^n$, $X^{-1}=\mathrm{diag}(x_1^{-1},\ldots,x_n^{-1})$, and $S^{-1}=\mathrm{diag}(s_1^{-1},\ldots,s_n^{-1})$. If $z\in\mathbb{R}^n$ is defined as $z=\mu X^{-1}e$, the optimality conditions of the problem $(\mathrm{P}')$ are

$$\begin{aligned} Ax&=b;\\ x+s&=u,\qquad x,s>0;\\ A^T y+z-w&=c,\qquad z,w>0;\\ SWe&=\mu e;\\ XZe&=\mu e,\end{aligned}\tag{2.1}$$

the equations in (2.1) are an implicit parameterization of *the central path*, [21].

In order to obtain the search direction $\Delta X = (\Delta x, \Delta s, \Delta y, \Delta z, \Delta w)^T$ , using (2.1) consider the maps $F$ given by

$$F(x,s,y,w,z) = \left(Ax-b,\, x+s-u,\, A^T y+z-w-c,\, XZe-\mu e,\, SWe-\mu e\right),$$

apply the Newton's method in the maps $F$ implies to solve the following linear system:

$$\begin{pmatrix} A & 0 & 0 & 0 & 0 \\ I_n & I_n & 0 & 0 & 0 \\ 0 & 0 & A^T & -I_n & I_n \\ Z & 0 & 0 & X & 0 \\ 0 & W & 0 & 0 & S \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta z \\ \Delta w \end{pmatrix} = \begin{pmatrix} r_b \\ r_u \\ r_c \\ r_1 \\ r_2 \end{pmatrix}, \tag{2.2}$$

where $r_b = b - Ax$, $r_u = u - x - s$, $r_c = c + w - z - A^T y$, $r_1 = \mu e - XZe$, $r_2 = \mu e - SWe$, $e^T = (1,\ldots,1) \in \mathbb{R}^n$, $X = \text{diag}(x_1,\ldots,x_n)$, $Z = \text{diag}(z_1,\ldots,z_n)$, $S = \text{diag}(s_1,\ldots,s_n)$, $W = \text{diag}(w_1,\ldots,w_n)$.

The predictor-corrector method modifies the right-hand side in (2.2) by

$$r_1 = \sigma\mu e - XZe, \quad r_2 = \sigma\mu e - SWe,$$

where $\sigma \in [0,1]$ is known as *centering parameter*.

Substituting the variables

$$\Delta s = r_u - \Delta x, \quad \Delta z = X^{-1}(r_1 - Z\Delta x) \quad \text{and} \quad \Delta w = S^{-1}(r_2 - W\Delta s)$$

in the third equation in (2.2) we have:

$$A^T\Delta y - (X^{-1}Z + S^{-1}W)\Delta x = r_c - X^{-1}r_1 + Sr_2 - S^{-1}Wr_u. \tag{2.3}$$

Considering the equation in (2.3) and the first equation in the system (2.2), we obtain the Augmented System

$$\begin{pmatrix} -\Theta^{-1} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} r \\ h \end{pmatrix}, \tag{2.4}$$

where $\Theta^{-1} = X^{-1}Z + S^{-1}W$, $\quad r = r_c - X^{-1}r_1 + S^{-1}r_2 - S^{-1}Wr_u$ and $h = r_b$. Substituting $\Delta x = \Theta A^T\Delta y - \Theta r$ in the equation $A\Delta x = h$ we obtain the Normal Equation's system

$$A\Theta A^T\Delta y = h + A\Theta r, \tag{2.5}$$

this system is symmetric and positive definite. In the next section, a Hybrid Preconditioner (HP) is used for the preconditioning of the matrix in (2.5).

## 3 PRECONDITIONER FOR THE NORMAL EQUATION SYSTEM

The preconditioning technique has as main objective to facilitate the convergence of iterative methods in order to find the solution of linear systems.

In this paper, two-sided preconditioning is used, that is, a preconditioner given by $K = K_1 K_2$. The iterative method is applied to

$$K_1^{-1} A K_2^{-1} z = K_1^{-1} b \quad \text{instead of} \quad Ax = b\,, \tag{3.1}$$

where $x = K_2^{-1} z$. Observe that, in (3.1) the system $Ax = b$ represents that system in (2.5) and the matrices $K_1$ and $K_2$ are the Controlled Cholesky Factorization (CCF) preconditioner in the first phase and the Splitting Preconditioner (SP) in the second phase. More precisely, in this paper, we denote as the Hybrid Preconditioner (HP) for the Interior Point Method (IPM) to the approch that computing the search direction via the Conjugate Gradient Method (CGM) in two phases, in the early iterations it uses the CCF preconditioner and after a phase change criteria the SP is used. The early iterations use the CCF preconditioner proposed in [4] successfully, but its performance reduces as the IPM approaches the optimal solution. This can be justified by the fact that the condition number of the non-preconditioned Normal Equation System, see equation (2.5), is of the order $O(\mu^{-2})$, where $\mu$ denotes the duality gap of the Linear Programming (LP) Problem, see [10]. However, the SP uses this characteristic to your favor to contain the difficulty provided of ill conditioning. We will make a description of the preconditioners used in this paper.

### 3.1 Controlled Cholesky Factorization preconditioner

Preconditioners based on an Incomplete Cholesky Factorizaton (ICF) present good performance if the fill-in of the preconditioning matrix is controlled, see [14]. Consider the matrix $A\Theta A^T$ from (2.5), let be $\mathscr{L}$ and $\tilde{\mathscr{L}}$, the lower triangular matrices from Cholesky factorization and Incomplete Cholesky Factorization (ICF) of matrix $A\Theta A^T$, respectively, that is,

$$\mathscr{L}\mathscr{L}^T = A\Theta A^T = \tilde{\mathscr{L}}\tilde{\mathscr{L}}^T + R\,,$$

where $R$ is the residual matrix. We define the matrix $E = \mathscr{L} - \tilde{\mathscr{L}}$, then,

$$\tilde{\mathscr{L}}^{-1}\left(A\Theta A^T\right)\tilde{\mathscr{L}}^{-T} = (I + \tilde{\mathscr{L}}^{-1}E)(I + \tilde{\mathscr{L}}^{-1}E)^T\,,$$

observe that if $\tilde{\mathscr{L}} \approx \mathscr{L}$ then $E \approx 0$ and therefore $\tilde{\mathscr{L}}^{-1}\left(A\Theta A^T\right)\tilde{\mathscr{L}}^{-T} \approx I_m$, this fact motivate the construction of the Controlled Cholesky Factorization (CCF) preconditioner. More precisely, the CCF preconditioner is a type of the ICF based in the minimization of the Frobenius norm of the matrix $E$, that is: $\min \quad \|E\|_{\mathrm{F}}^2$. For this is consideraded the minimize problem:

$$\min \sum_{j=1}^{m} c_j, \;\; \text{where} \;\; c_j = \sum_{i=1}^{m} |l_{ij} - \tilde{l}_{ij}|^2\,, \tag{3.2}$$

rewriting this problem, we have:

$$\min \quad \sum_{j=1}^{m} \left( \sum_{k=1}^{m_j+\eta} |l_{i_k j} - \tilde{l}_{i_k j}|^2 + \sum_{k=m_j+\eta+1}^{m} |l_{i_k j}|^2 \right), \tag{3.3}$$

where $m$ is the order of the matrix, $m_j$ is the number of nonzero entries below the diagonal in the $j$-th column of the matrix $A\Theta A^T$ and $\eta$ is the extra number of nonzero entries allowed per

column. Note that we want to minimize the problem in (3.3), the entries $\tilde{l}_{i_k j}$ will be higher in absolute value. We will denote by $\hat{\mathscr{L}}$ the CCF matrix, that is, the matrix containing the major elements of ICF. Thus, the NES given in 3.4 preconditioned by CCF preconditioner is:

$$\hat{\mathscr{L}}^{-1} A\Theta A^T \hat{\mathscr{L}}^{-T} \Delta\hat{y} = \hat{\mathscr{L}}^{-1}(h + A\Theta r), \tag{3.4}$$

where $\Delta\hat{y} = \hat{\mathscr{L}}^T \Delta y$.

In the first iteration the number of nonzero entries allowed is given by:

$$\eta_0 = \begin{cases} \operatorname{nnz}(A\Theta A^T)/m, & \text{if } \operatorname{nnz}(A\Theta A^T) < 10m; \\ -\operatorname{nnz}(A\Theta A^T)/m, & \text{other case.} \end{cases} \tag{3.5}$$

As the number of CGM iterations is increased, it becomes necessary that the value of $\eta$ is increased. That is, if the number of CGM iterations exceeds $m/5$ the value of $\eta$ is increased by 10, see [3]. When $\eta > 10$ the change of phases is made in Hybrid Preconditioner (HP), see [20].

In the construction of the CCF preconditioner it is possible to find diagonal faults, these faults are corrected with an exponential increase. The increment value is:

$$\alpha_t = 5 \cdot 10^{-4} \cdot 2^{t-1}, \tag{3.6}$$

where $t = 1, \ldots, 15$ represents the number of allowable restarts on the CCF, see [3]. There are other sequences to compute an increase in the diagonal, see [11, 12, 15].

In this way, every time that a diagonal fault occurs the computing of the matrix elements $\hat{\mathscr{L}}$ is restarted and if the number of restarts is 15, the value of $\alpha$ is not a small value. In order to avoid restarts an approach is proposed in [18]. This approach was based on the paper of [2]. In Section 4, we apresented an approach such that the number of restarts to compute the CCF preconditioner is reduced by looking for the value of the increment in the main diagonal of matrix $A$ to be close to the value proposed by [15].

### 3.2 Splitting preconditioner applied to the normal equation system

The Splitting Preconditioner is based on the complementary slackness conditions of the Linear Programming problem (P) and (D), that is,

$$x_i z_i = 0 \quad \text{and} \quad s_i w_i = 0 \quad \text{for all} \quad i = 1, \ldots, n. \tag{3.7}$$

Note that the diagonal matrix $\Theta$ components $\theta_j = (z_j/x_j + w_j/s_j)^{-1}$ given in (2.4) and (2.5) changes in each IPM iteration, particularly next to the optimal solution due to (3.7) and the non-negativity of the variables $x, z, s, w$, there will be indexes $j \in \{1, \ldots n\}$ such that $\theta_j \to 0$ or $\theta_j \to \infty$. This feature is the reason for the good performance of the Splitting preconditioner in the last IPM iterations.

In each IPM iteration consider the ordering $\theta_{\sigma(1)} \geq \ldots \geq \theta_{\sigma(m)} \geq \ldots \geq \theta_{\sigma(n)}$, where $\sigma$ is a permutation of the set $\{1, \ldots n\}$, this permutation changes from iteration to iteration. The sets of

indexes are denoted by $\mathscr{B} = \{\sigma(1), \ldots, \sigma(m)\}$ and $\mathscr{N} = \{\sigma(m+1), \ldots, \sigma(n)\}$, if the $A$ and $\Theta$ columns are reordered according to $\sigma$, the matrix (2.5) can be written as

$$A\Theta A^T = A_{\mathscr{B}}\Theta_{\mathscr{B}}A_{\mathscr{B}}^T + A_{\mathscr{N}}\Theta_{\mathscr{N}}A_{\mathscr{N}}^T . \tag{3.8}$$

If the submatrix $A_{\mathscr{B}}$ is non-singular, the Splitting preconditioner for the normal equation's system is given by the matrix

$$P = A_{\mathscr{B}}\Theta_{\mathscr{B}}^{1/2}, \tag{3.9}$$

in this case, $\mathscr{B}$ and $A_{\mathscr{B}}$ are known as basic indexes and base of the SP , respectively.

Preconditioning the matrix given in (3.8) for $P$, we obtain

$$P^{-1}(A\Theta A^T)P^{-T} = I_m + WW^T \text{where } W = \Theta_{\mathscr{B}}^{-1/2}A_{\mathscr{B}}^{-1}A_{\mathscr{N}}\Theta_{\mathscr{N}}^{1/2}.$$

An ideal situation would occur if $\Theta_{\mathscr{B}}^{-1/2} \to 0$ and $\Theta_{\mathscr{N}}^{1/2} \to 0$ implying that $W \to 0$ and, thus, $P^{-1}(A\Theta A^T)P^{-T} \approx I_m$. However, nothing guarantees that this matrix $A_{\mathscr{B}}$ is non-singular and even if so, not all $\theta_j$ with $j \in \mathscr{B}$ is a large value. In fact, close to the optimal solution there are at least $n-m$ values close to zero, this implies that there will be a maximum of $m$ not small values.

However, if $\mathscr{B} = \{\sigma(1), \ldots, \sigma(m)\}$ is the indexes set used in an IPM iteration for the SP construction, an advantageous property of this preconditioner is that the same indexes may be reused by several iterations making these iterations much cheaper. The papers [1, 5, 9, 19] study the choice of basic indexes for a preconditioner.

In order to study the condition number of the preconditioned matrix, assume that $\lambda$ and $v$ are an eigenvalue and an eigenvector of the matrix $I + WW^T$, that is, $v + WW^T v = \lambda v$, multiplying by vector $v^T$ this equation, we note that $|\lambda| \geq 1$, that is

$$\kappa(P^{-1}(A\Theta A^T)P^{-T}) = \frac{\lambda_{\max}}{\lambda_{\min}} \leq \lambda_{\max} .$$

On the other hand, we observe that: $\lambda_{\max}\left(P^{-1}A\Theta A^T P^{-T}\right) = \|P^{-1}A\Theta^{1/2}\|_2^2$ and

$$\lambda_{\max}\left(P^{-1}A\Theta A^T P^{-T}\right) =\leq \|P^{-1}A\Theta^{1/2}\|_F^2 = \sum_{i=1}^{n} \theta_j \|P^{-1}A_j\|_2^2 , \tag{3.10}$$

we use (3.10) to find an upper bound condition number $\kappa(P^{-1}(A\Theta A^T)P^{-T})$ in the Section 4.3.

To find the linearly independent columns of $A$ to form the base of the SP may require an excessive memory usage because it is done trough a factorization LU of matrix $A$, a small pivot or zero indicates that the column corresponding is linearly dependent. The technique proposed in [17] to deal the excessive fill-in is the interruption of the factorization and reordering the independent columns found thus far by the number of nonzero entries. In [17], the authors of the SP suggested the choice of the first $m$ linearly independent columns of matrix $A$ reordered giving priority to the indexes $j$ of the $\theta_j/\|A_j\|_1$ values in decreasing order. In [20] was proposed a new column ordering according $\theta_j/\|A_j\|_2$ values in decreasing order, which achieved better results in the SP performance.

## 4 NEW PROPOSALS

### 4.1 Fault correction parameter

Hereafter, the matrix $A\Theta A^T$ is denoted by $\mathscr{A}$. Suppose $\mathscr{A}$ is a scaled matrix, that is, $a_{jj} = 1$ and $a_{ij} \leq 1$ for $i, j = 1, \ldots m$. In the construction of the CCF preconditioner, it is said that there is a diagonal fault when $d_j < \varepsilon$ for some $j = 1, \ldots, m$.

The proposal for the computation of the new increment $\alpha_t$ considers the $LDL^T$ factorization of $\mathscr{A}$ and $\mathscr{A} + \alpha I$. That is, if $\overline{\mathscr{A}} = \mathscr{A} + \alpha I$, we look for the matrices $L$, $D$, $\overline{L}$ and $\overline{D}$ such that $\overline{\mathscr{A}} = \overline{L}\,\overline{D}\,\overline{L}^T$. The subscript $t$ of $\alpha_t$ indicates the number of attempts to correct the diagonal fault. The CCF preconditioner allows up to fifteen attempts, that is, up to fifteen restarts in its construction. From now on, for simplicity, $\alpha_t$ is denoted only as $\alpha$. Next, we establish the dependence between the entries of the matrices $\overline{L}$ and $\overline{D}$ with respect to the parameter $\alpha$:

$$\overline{d}_j = \overline{a}_{jj} - \sum_{k=1}^{j-1} \overline{d}_k \overline{\ell}_{jk}^2 ; \tag{4.1a}$$

$$\overline{\ell}_{ij} = \frac{1}{\overline{d}_j} \left( \overline{a}_{ij} - \sum_{k=1}^{j-1} \overline{\ell}_{ik} \overline{d}_k \overline{\ell}_{jk} \right) ; \tag{4.1b}$$

given that $\overline{\mathscr{A}} = \mathscr{A} + \alpha I$, we get:

$$\overline{d}_j = a_{jj} + \alpha - \sum_{k=1}^{j-1} \overline{d}_k \overline{\ell}_{jk}^2 ; \tag{4.2a}$$

$$\overline{\ell}_{ij} = \frac{1}{\overline{d}_j} \left( a_{ij} - \sum_{k=1}^{j-1} \overline{\ell}_{ik} \overline{d}_k \overline{\ell}_{jk} \right) , \tag{4.2b}$$

for $j = 1, \ldots m$ and $i = j+1, \ldots, m$. We obtain $\overline{d}_j > \varepsilon$ in (4.2a) each time a value $\alpha$ is incremented in the diagonal of $\mathscr{A}$, this would imply that the numerical value of the sumatory $\sum_{k=1}^{j-1} \overline{d}_k \overline{\ell}_{jk}^2$ is decreasing. Computationally, this verified in [4] and this study presented the Proposition 4.1, to justify this fact.

**Proposition 4.1.** *If $d_j < \varepsilon$, the functions $F_u : \mathbb{R}^+ \longrightarrow \mathbb{R}$, given by*

$$\alpha \longmapsto \sum_{k=1}^{u-1} \overline{d}_k \overline{\ell}_{uk}^2 , \tag{4.3}$$

*are decreasing, where $u = 2, \ldots, j$ and $F_u(0) = \sum_{k=1}^{u-1} d_k \ell_{uk}^2$. Furthermore,*

*(i) $\overline{d}_j > \varepsilon$ for all $\alpha \geq \varepsilon - d_j$.*

*(ii) If $0 < d_j < \varepsilon$, then $\overline{d}_j > \varepsilon$ for all $\alpha \geq \varepsilon$.*

**Proof.** Let's $j$ such that $d_j < \varepsilon$ and we assume that the function $F_j$ is decreasing and we proof this fact later. So for every $\alpha > 0$, we have $F_j(0) > F_j(\alpha)$ or

$$\sum_{k=1}^{j-1} \left( d_k \ell_{jk}^2 - \overline{d}_k \overline{\ell}_{jk}^2 \right) > 0 . \tag{4.4}$$

This inequality is used to show ((i)) and ((ii)):

(i) In fact, by equation (4.2a):

$$\begin{aligned}\overline{d}_j &= a_{jj} - \sum_{k=1}^{j-1}\left(d_k\ell_{jk}^2\right) + \alpha + \sum_{k=1}^{j-1}\left(d_k\ell_{jk}^2\right) - \sum_{k=1}^{j-1}\overline{d}_k\overline{\ell}_{jk}^2 \\ &= d_j + \alpha + \sum_{k=1}^{j-1}\left(d_k\ell_{jk}^2 - \overline{d}_k\overline{\ell}_{jk}^2\right) > d_j + \alpha \geq \varepsilon\,,\end{aligned} \tag{4.5}$$

the first inequality follows from (4.4) and the second inequality from $\alpha \geq \varepsilon - d_j$.

(ii) Observe that $\alpha \geq \varepsilon$, we have:

$$\overline{d}_j = d_j + \alpha + \sum_{k=1}^{j-1}\left(d_k\ell_{jk}^2 - \overline{d}_k\overline{\ell}_{jk}^2\right) > d_j + \alpha \geq d_j + \varepsilon > \varepsilon\,, \tag{4.6}$$

since $\alpha \geq \varepsilon$ and $d_j > 0$.

In this way, the next step is to prove that the function $F_u$ defined in (4.3), is decreasing for $u = 2,\ldots,j$ where $j = 2,\ldots,m$.

For every $j$, differentiating the function $\overline{d}_k\overline{\ell}_{jk}$ in relation to the variable $\alpha$:

$$(\overline{d}_k\overline{\ell}_{jk})' = \overline{d}_k'\overline{\ell}_{jk} + \overline{d}_k\overline{\ell}_{jk}'\,. \tag{4.7}$$

Since that $F_j'(\alpha) = \sum_{k=1}^{j-1}\left(\overline{d}_k'\overline{\ell}_{jk}^2 + 2\overline{\ell}_{jk}(\overline{d}_k\overline{\ell}_{jk}')\right)$, we use the Equation 4.7 to obtain:

$$F_j'(\alpha) = \sum_{k=1}^{j-1}\left(-\overline{d}_k'\overline{\ell}_{jk}^2 + 2\overline{\ell}_{jk}(\overline{d}_k\overline{\ell}_{jk})'\right). \tag{4.8}$$

The following will demonstrate that for every $u = 2,\ldots,j$, $F_t'(\alpha) < 0$ and consequently $F_j$ will be decreasing. In fact, using Mathematical Induction,

a) **Basis**: when $u = 2$, from the equation (4.1):

$$\overline{d}_1' = (d_1 + \alpha)' = 1 \quad \text{and} \quad (\overline{d}_1\overline{\ell}_{21})' = (d_1\ell_{21})' = 0\,. \tag{4.9}$$

Substituting the equations from 4.9 in (4.8):

$$F_2'(\alpha) = -\overline{d}_1'\overline{\ell}_{21}^2 + 2\overline{\ell}_{21}(\overline{d}_1\overline{\ell}_{21})' = -\overline{\ell}_{21}^2 \leq 0\,.$$

b) **Inductive step**: Assume for every $u = 2,\ldots,j-1$, $F_u'(\alpha) \leq 0$. It must then be shown that $F_j'(\alpha) \leq 0$. Otherwise, assume $F_j'(\alpha) > 0$, from the equation (4.8):

$$0 < F_j'(\alpha) = \sum_{k=1}^{j-1}\left(-\overline{d}_k'\overline{\ell}_{jk}^2 + 2\overline{\ell}_{jk}(\overline{d}_k\overline{\ell}_{jk})'\right),$$

since that $0 \leq \sum_{k=1}^{j-1} \overline{d}_k' \overline{\ell}_{jk}^2$, we have:

$$0 < \sum_{k=1}^{j-1} 2\overline{\ell}_{jk}(\overline{d}_k \overline{\ell}_{jk})'. \tag{4.10}$$

If the inequality (4.10) were true, it would imply that exist an index $r$, $r \in \{1,\ldots,j-1\}$, such that $\overline{\ell}_{jr}(\overline{d}_r\overline{\ell}_{jr})' > 0$. From the equation (4.10), it would follow that the function $(\overline{d}_r\overline{\ell}_{jr})^2$ is increasing. In fact, for all $u \in \{1,\ldots,j-1\}$ the function $\overline{d}_u$ is increasing, since that we have the equation (4.1a) and the basis of the Mathematical Induction:

$$\overline{d}_u' = 1 - \sum_{k=1}^{u-1} (\overline{d}_k \overline{\ell}_k^2)' = 1 - F_u'(\alpha) > 0\,, \tag{4.11}$$

for every $u = 1,\ldots,j-1$. Thus, when $\alpha > 0$, we use (4.11) to obtain: [1]

$$\overline{d}_u > d_u > \varepsilon > 0\,,$$

that mean, $\overline{d}_t > 0$ for all $u \in \{1,\ldots,j-1\}$. In order to prove that the function $(\overline{d}_r\overline{\ell}_{jr})^2$ is increasing, we must consider $u = r$ to garantee $\overline{d}_r > 0$. Thus, from the equation (4.10):

$$\left((\overline{d}_r\overline{\ell}_{jr})^2\right)' = 2\overline{d}_r\overline{\ell}_{jr}(\overline{d}_r\overline{\ell}_{jr})' > 0\,, \tag{4.12}$$

that is, the function $(\overline{d}_r\overline{\ell}_{jr})^2$ is increasing. We use (4.12) in order to obtain a contradiction. If $(\overline{d}_r\overline{\ell}_{jr})^2$ is increasing, for all $\alpha > 0$,

$$(d_r\ell_{jr})^2 < (\overline{d}_r\overline{\ell}_{jr})^2\,,$$

and consequently from the equation (4.11), for all $\alpha > 0$, it would follow that:

$$(d_r\ell_{jr})^2 < (\overline{d}_r\overline{\ell}_{jr})^2 = \left(d_r\ell_{jr} + \sum_{s=1}^{r-1} \ell_{js}d_s\ell_{rs} - \overline{\ell}_{js}\overline{d}_s\overline{\ell}_{rs}\right)^2.$$

However, when $\alpha$ approaches zero:

$$(d_r\ell_{jr})^2 < \lim_{\alpha \longrightarrow 0} \left(d_r\ell_{jr} + \sum_{s=1}^{r-1} \ell_{js}d_s\ell_{rs} - \overline{\ell}_{js}\overline{d}_s\overline{\ell}_{rs}\right)^2 = (d_r\ell_{jr})^2\,,$$

that means, a contradiction.

Therefore, $F_u'(\alpha) \leq 0$ for every $u = 2,\ldots,j$. Since that $j$ is arbitrary we have that the function $F_j$ is decreasing, for every $j = 2,\ldots,m$. □

[1] Observe that if exist diagonal fault in $j$-column, $d_u > \varepsilon$ for all $u = 1,\ldots,j-1$.

As a consequence of the Proposition 4.1, the value $\alpha = \varepsilon - d_j$ could be used. However, it is necessary that $\alpha$ be as small as possible so that in this way $\overline{\mathscr{A}} \approx \mathscr{A}$ and we have nothing to guarantee that $\varepsilon - d_j$ is a small value. Thus, it is proposed to solve the following problem:

$$(\overline{\mathrm{P}}_\alpha) \begin{cases} \min\limits_{\alpha>0} & \alpha \\ \text{s.t.} & \sum\limits_{k=1}^{j-1} \overline{d}_k \overline{\ell}_{jk}^2 \leq a_{jj} + \alpha - \varepsilon\,. \end{cases} \tag{4.13}$$

This approach is a consequence of the equation (4.1) and the fact that for each $\alpha > 0$, $\overline{d}_j > \varepsilon$ is satisfied, if, and only if, $\sum_{k=1}^{j-1} \overline{d}_k \overline{\ell}_{jk}^2 \leq a_{jj} + \alpha - \varepsilon$. Observe that in this case, the values $\overline{d}_k$ and $\overline{\ell}_{jk}$ are not known for all $k = 1, \ldots, j-1$, because the factorization for obtain $\overline{L}\overline{D}\overline{L}^T$, has not been done yet. In order to get an approximation of the solution of the problem $(\overline{\mathrm{P}}_\alpha)$, look for a function that is equivalent to $F_j : \mathbb{R} \to \mathbb{R}$, given by $\alpha \mapsto \sum_{k=1}^{j-1} \overline{d}_k \overline{\ell}_{jk}^2$ when $\alpha$ approaches zero[2], for some $j \in \{2, \ldots m\}$.

Using the Proposition 4.1, for every $\alpha > 0$, we have $F_j(\alpha) < F_j(0)$ or

$$\sum_{k=1}^{j-1} (\overline{d}_k \overline{\ell}_{tk}^2) < \sum_{k=1}^{j-1} (d_k \ell_{jk}^2)\,, \tag{4.14}$$

where $j = 2 \ldots, m$. Consider the following functions

$$\begin{array}{lll} f_j : \mathbb{R} \to \mathbb{R} & \text{and} & g_j : \mathbb{R} \to \mathbb{R} \quad \text{given by} \\ \alpha \mapsto \sum\limits_{k=1}^{j-1} \dfrac{(d_k \ell_{jk})^2}{d_k + \alpha} & & \alpha \mapsto \sum\limits_{k=1}^{j-1} \dfrac{\alpha}{d_k + \alpha} d_k \ell_{jk}^2\,, \end{array} \tag{4.15}$$

respectively. We use the functions $f_j$ and $g_j$ because for every $\alpha > 0$, we have:

$$(f_j + g_j)(\alpha) = \sum_{k=1}^{j-1} d_k \ell_{jk}^2\,;$$

furthermore, from the Proposition 4.1, we obtain: $\sum_{k=1}^{j-1} \overline{d}_k \overline{\ell}_{jk}^2 \leq (f_j + g_j)(\alpha)$.

Since that $f_j(\alpha) \sim \sum_{k=1}^{j-1} \overline{d}_k \overline{\ell}_{jk}^2$, when $\alpha$ approaches zero, we are looking for the solution of the following problem:

$$(\mathrm{P}_\alpha) \begin{cases} \min\limits_{\alpha>0} & \alpha \\ \text{s.t.} & f_j(\alpha) \leq a_{jj} - \varepsilon + \alpha\,, \end{cases}$$

in order to obtain an approximate solution of $(\overline{\mathrm{P}}_\alpha)$.

Since the function $f_j$ is decreasing, we have that $\alpha$ is solution of the problem $(\mathrm{P}_\alpha)$ if, and only if, $f_j(\alpha) = a_{jj} - \varepsilon + \alpha$. We use the Newton Raphson method for computing the numerical value of $\alpha$ in the Algorithm 1.

[2] The functions $f(x)$ and $g(x)$ are are called equivalents when $x$ approaches $a$ if $\lim_{x \longrightarrow a} \frac{f(x)}{g(x)} = 1$; this is denoted as $f \underset{x \to a}{\sim} g$.

**Algorithm 1:** Find the value $\alpha$ that solve the equation $f_j(\alpha) = a_{jj} - \varepsilon + \alpha$.

**input :** The index $j$ such that $d_j < \varepsilon$, function $f_j$, entry $a_{jj}$ and tolerance
**output:** $\alpha$.

**Function** `NewtonRaphson(`$j$, $f_j$,$a_j$, $\varepsilon$, tolerance`):`
  $i \leftarrow 0$;
  $\alpha_0 = 0$;
  **while** $|f_j(\alpha_i)| >$ tolerance **do**
    $i \leftarrow i+1$;
    $\alpha_i \leftarrow \alpha_{i-1} - \dfrac{1}{f_j(\alpha_{i-1}) - a_{jj} + \varepsilon - \alpha_{i-1}}(f'(\alpha_{i-1}) - 1)$;
    $\alpha \leftarrow \alpha_i$;
  **return** $\alpha$

### 4.2 Modification in the fill-in of the Controlled Cholesky Factorization preconditioner

We will denote by $\eta$ the fill-in parameter of the Controlled Cholesky Factorization (CCF) preconditioner. The objective in this proposal is to ensure that the number of nonzero entries (nnz) of the matrix $\hat{\mathscr{L}}$ in the equation (3.4) is at most $(\text{nnz}(\mathscr{A}) + 3m)/2$ when the CCF preconditioner is used in the IPM iterations. Thus, from (3.4) we have $\text{nnz}(\hat{\mathscr{L}}) = \text{nnz}(\tilde{\mathscr{L}})$.

In order to determine the initial parameter $\eta$, denoted by $\eta_0$, the quotient $\text{nnz}\mathscr{A}/\text{nnz}A$ is computed and from it we have the following cases:

(i) $\eta_0 = 1$, if $1 \leq \text{nnz}\mathscr{A}/\text{nnz}A < 2$; (ii) $\eta_0 = -\text{nnz}A/m$, otherwise.

If the number of the preconditioned CGM iterations is greater than $m/5$, the heuristic to determine the increment of $\eta$ is given by:

(i) $\eta_k = 1$, when $\eta_0 = 1$;

(ii) If for all $j = 1,\ldots,m$ and $i = j+1,\ldots,m$, $|\bar{\ell}_{ij}| < 1/1+\alpha$ in the iteration $k-1$, then the value $\eta_k$ will be incremented, $\eta_k = \eta_{k-1}/2$, if $\eta_{k-1} < 0$.

The final $\eta$, denoted by $\eta_f$, in both cases ((i)) and ((ii)) will be $\eta_f \leq 1$. Thus, the largest fill-in allowed for $\hat{\mathscr{L}}$ will be $(\text{nnz}\mathscr{A} + m)/2 + m$.

### 4.3 New ordering criteria of basic indexes for Spliting Preconditioner

Based in the observations presented in the Section 3.2, the idea arises of an ordering that simultaneously considers well conditioning and sparsity for the base of the Splitting Preconditioner (SP). We denote by $\text{nnz}(A_j)$ to the number of nonzero entries in column $A_j$ of the constrained matrix $A_{m\times n}$ of the LP problem, where for $j = 1,\ldots,n$.

Observe that; $1 \leq \text{nnz}(A_j) \leq m$ for all column $A_j$ of $A$, however, in sparse problems $\text{nnz}(A_j) << m$. Define $k_j = \theta_j^{1/2}/\text{nnz}(A_j)$ and perform a decreasing ordering of the $k_j$ elements, with this order the Algorithm 2 is proposed.

**Algorithm 2:** In order to find the basic indexes of the Splitting preconditioner

**input :** The matrix $A \in \mathbb{R}^{m\times n}$ with rank $m$ and the diagonal matrix $D$.
**output:** The basic and non-basic index sets $\mathscr{B} = \{b_1, \ldots b_m\}$, $\mathscr{N} = \{1,\ldots,n\}\backslash\mathscr{B}$.

**begin**
  Find the permutation $\sigma$ of the set $\{1,\ldots,n\}$ such that: $k_{\sigma(1)} \geq k_{\sigma(2)} \geq \ldots \geq k_{\sigma(n)}$;
  Define $\mathscr{B} = \emptyset$, $i = 1$, $k = 0$;
  **while** $|\mathscr{B}| < m$ **do**
    **if** $A_{\sigma(i)}$ *is linearly independent to* $\{A_j : j \in \mathscr{B}\}$ **then**
      $\mathscr{B} = \mathscr{B} \cup \{\sigma(i)\}$ ; $k = k+1$; $b_k = \sigma(i)$;
    $i = i+1$.

The non-increasing ordering of values $k_j's$ is motivated by the following reason: if two columns $A_{j_1}$ and $A_{j_2}$ satisfy $\text{nnz}(A_{j_1}) \leq \text{nnz}(A_{j_2})$, that is $A_{j_1}$ is more sparse than $A_{j_2}$, then,

$$1/\text{nnz}(A_{j_1}) \geq 1/\text{nnz}(A_{j_2}).$$

Therefore, the column $A_{j_1}$ will have priority over $A_{j_2}$ if $\theta_{j_1} \approx \theta_{j_2}$. Thus, while the values $\theta_j^{1/2}$ will be used in the Theorem 4.1 to take care of well conditioning, the values $\text{nnz}(A_j)$ will give priority to the sparse columns. The algoritm 2 and the proof of the Theorem 4.1 are based in [16] adding a condition that takes into account the sparseness of the $A$ columns. To simplify the notation we consider the permutation $\sigma = id$, where $id$ is the identity permutation, in addition, we denote $A_{\mathscr{B}}$ simply by $B$.

**Teorema 4.1.** *Suppose that the basic and non-basic index sets $\mathscr{B}$ and $\mathscr{N}$ of the Splitting preconditioner are obtained by the algorithm 2. Then*

1. $\theta_j^{1/2}\|\Theta_{\mathscr{B}}^{-1/2}B^{-1}A_j\| = 1$ *for* $j \in \mathscr{B}$;
2. $\theta_j^{1/2}\|\Theta_{\mathscr{B}}^{-1/2}B^{-1}A_j\| \leq \text{nnz}(A_j)\|B^{-1}A_j\|$ *for* $j \in \mathscr{N} = \{1,\ldots,n\}\backslash\mathscr{B}$. *Also,* $\kappa(P^{-1}A\Theta A^T P^{-T}) \leq nK^2\|B^{-1}A\|^2$, *when* $K = \max\{\text{nnz}(A_j) : j = 1,\ldots n\}$.

**Proof.** The proof this theorem consider two cases.

Case 1. If $j \in \mathscr{B}$, then $B^{-1}A_j = e_j$ where $e_j$ is the $j$-th unit vector of $\mathbb{R}^m$ so,

$$\theta_j^{1/2}\|\Theta_{\mathscr{B}}^{-1/2}B^{-1}A_j\| = \theta_j^{1/2}\|\Theta_B^{-1/2}e_j\| = \theta_j^{1/2}\|\theta_j^{-1/2}e_j\| = 1. \tag{4.16}$$

Case 2. If $j \in \mathscr{N}$, two cases are considered.

Case 2.1. The column $A_j$ was not considered to enter the base according to Algorithm 2, that is $j > b_i$, thus

$$k_{b_i} \geq k_j \quad \text{for all} \quad b_i \in \mathscr{B}. \tag{4.17}$$

Let $\theta_0^{1/2} = \min\{\theta_{b_i}^{1/2} : b_i \in \mathscr{B}\}$, if $k_0 := \theta_0^{1/2}/\operatorname{nnz}(A_0)$, since $b_m$ is the last basic index we have $k_0 \geq k_{b_m}$, using (4.17) $k_0 \geq k_j$, thus

$$\begin{aligned} \theta_j^{1/2}\|\Theta_{\mathscr{B}}^{-1/2}B^{-1}A_j\| &\leq \frac{d_j^{1/2}\|B^{-1}A_j\|}{\min\{\theta_{b_i}^{1/2} : b_i \in \mathscr{B}\}} \\ &= \frac{k_j \operatorname{nnz}(A_j)}{k_0 \operatorname{nnz}(A_0)}\|B^{-1}A_j\| \\ &\leq \operatorname{nnz}(A_j)\|B^{-1}A_j\|. \end{aligned} \tag{4.18}$$

Case 2.2. The column $A_j$ was considered to be $r$-th column of $B$, however $A_j$ resulted to be linearly dependent on the colunms $A_{b_1}, A_{b_2}, \ldots, A_{b_{r-1}}$, that is, $A_j = B[u,0]^T$, where $u \in \mathbb{R}^{r-1}$, observe that $\|u\| = \|B^{-1}A_j\|$. Furthermore, $k_{b_i} \geq k_j$ for all $i = 1, \ldots, r-1$. If $\theta_0^{1/2} = \min\{\theta_{b_1}^{1/2}, \ldots, \theta_{b_{r-1}}^{1/2}\}$ and we define $k_0 := \theta_0^{1/2}/\operatorname{nnz}(A_0)$, then $k_0 \geq k_{b_{r-1}} \geq k_j$, thus,

$$\begin{aligned} \theta_j^{1/2}\|\Theta_{\mathscr{B}}^{-1/2}B^{-1}A_j\| &= \theta_j^{1/2}\left(\sum_{i=1}^{r-1}\theta_{b_i}^{-1}u_i^2\right)^{1/2} \\ &\leq \frac{k_j \operatorname{nnz}(A_j)}{k_0 \operatorname{nnz}(A_0)}\|B^{-1}A_j\| \\ &\leq \operatorname{nnz}(A_j)\|B^{-1}A_j\|. \end{aligned} \tag{4.19}$$

Using (3.10) we have that

$$\lambda_{\max} = \|\Theta_{\mathscr{B}}^{-1/2}B^{-1}A\Theta^{1/2}\|_2^2 \leq \sum_{j=1}^{n}\theta_j\|\Theta_{\mathscr{B}}^{-1/2}B^{-1}A_j\|_2^2, \tag{4.20}$$

substituting (4.16), (4.18) and (4.19) in (4.20) we have

$$\begin{aligned} \lambda_{\max}\left(P^{-1}A\Theta A^T P^{-T}\right) &\leq K^2\sum_{j=1}^{n}\|B^{-1}A_j\|^2 = K^2\|B^{-1}A\|_F^2 \\ &\leq mK^2\|B^{-1}A\|^2. \end{aligned}$$

Furthermore, we have

$$\kappa(P^{-1}A\Theta A^T P^{-T}) \leq mK^2\|B^{-1}A\|^2, \tag{4.21}$$

since $\lambda_{\min}\left(P^{-1}A\Theta A^T P^{-T}\right) \geq 1$. □

Note that the condition number of the matrix preconditioned in (4.21) is uniformly bounded by amount that depends on the data of the problem and not on the interior point method iteration.

## 5 NUMERICAL EXPERIMENTS

The PCx was originally proposed by [6] and in this paper to perform the numerical experiments the PCx was modified. The direct method used in PCx to obtain the solution of linear systems was replaced by an iterative method [3].

The tests performed compare the Hybrid Preconditioner (HP) proposed in [20], and the preconditioner presented in this study, denoted by HPmod. In SP, the base $B$ can be maintained in some iterations of the IPM, this base is changed when $8 * n_g \geq m$, where $n_g$ denotes the number of iterations of preconditioned CGM in an IPM iteration. The problems used are in the public domain of the Netlib, QAP, and Kennington repositories.

The first two columns of Table 1 indicate the number of rows and columns of preprocessed problems. The number of CCF restarts corresponding to all iterations of the first phase, the time each problem was solved and measured in seconds, and finally, in the last two columns, the number of iterations of the IPM for solve each problem. For the comparison of approaches, the results presented in Table 1 can be summarized in performance profiles. These profiles use a logarithmic scale in the base 2, see [7].

Large scaled problems were tested and the criteria for choosing them depended on whether the number of rows or columns were greater than 5000. The most significant differences are in bold. The symbol "$-$" indicates that the problem has not been resolved, the symbol "$^{\ddagger}$" mean that the total number of restarts was greater than 15 in more than one iteration of the IPM.

The HPmod works well compared to the HP preconditioner when evaluating the total number of iterations, see Fig. 1. The proposal to use a hybrid approach is that the CCF and the SP did not work well on their own in most of the problems tested. According to [3], the CCF preconditioner shows good results in the initial iterations of the IPM, however it may deteriorate in the latter, since the matrix $\mathscr{A}$ becomes ill-conditioned. If the SP is used in the early iterations, it is possible that the optimal solution is not found, that happened in the problems: nug05-3rd, nug06-3rd, nug07-3rd and nug08-3rd. In particular, in these problems the HP changes phase, which does not happen in the HPmod, that is, at least in these problems it was not necessary to carry out the phase change and, therefore, the optimal solution was obtained by the CCF preconditioner. In the problems that there was a phase change, when we used the HPmod, the SP computation is performed in less time compared to SP proposed by [20], due to the sparse columns that were used.

Note that the CCF preconditioner that we propose can solve problems osa-14, osa-30 and osa-60 without restarts, see Table 1. Therefore, to elaborate the performance profile presented in Fig. 2 the value 0.1 was considered instead of 0. Using the preconditioner CCF the diagonal fault correction parameter allows to increase a value lower than that computed with the CCF.

The computation of the preconditioning CCF proposed by [4], works with a diagonal matrix when the number of restarts is greater than 15. Observe that preconditioning a system in the early iterations using a diagonal matrix seems a good strategy, however in iterations that are not

Table 1: Performance of the approaches.

| | Size | | Restarts | | Time | | Iteration | |
|---|---|---|---|---|---|---|---|---|
| Problem | Rows | Colums | HP | HPmod | HP | HPmod | HP | HPmod |
| maros-r7 | 2152 | 7440 | 173‡ | 13 | 16,24 | **8,90** | 30 | 22 |
| NL | 6665 | 14680 | 284 | 75 | 33,74 | **23,74** | 41 | 41 |
| BL | 5729 | 12462 | 261 | 100 | 18,05 | **15,10** | 38 | 38 |
| stocfor3 | 15362 | 22228 | 199 | 55 | 89,50 | **55,41** | 32 | 32 |
| els19 | 4350 | 13186 | 78 | 50 | 43,51 | **42,06** | 31 | 31 |
| chr22b | 5587 | 10417 | 79 | 38 | 19,10 | **18,19** | 29 | 29 |
| chr25a | 814 | 15325 | 64‡ | 37 | 38,60 | **36,35** | 29 | 28 |
| nug05-3rd | 1410 | 1425 | 20 | 25 | – | **0,24** | – | 6 |
| nug06-3rd | 3972 | 4686 | 30 | 37 | – | **5,92** | – | 7 |
| nug07-3rd | 9422 | 12691 | 30 | 40 | – | **32,29** | – | 8 |
| nug08-3rd | 19728 | 29856 | 30 | 47 | – | **208,34** | – | 9 |
| qap12 | 2794 | 8856 | 64‡ | 5 | – | **104,06** | – | 20 |
| scr15 | 2234 | 6210 | 64‡ | 43 | **6,49** | 6,58 | 24 | 24 |
| scr20 | 5079 | 15980 | 74 | 55 | 60,18 | **52,76** | 21 | 21 |
| rou20 | 7359 | 37640 | 81 | 43 | 754,32 | **658,29** | 24 | 24 |
| cre-a | 2989 | 6692 | 116‡ | 29 | 7,02 | **4,25** | 28 | 27 |
| cre-b | 5328 | 36382 | 288 | 147 | 43,33 | **37,79** | 43 | 43 |
| cre-c | 2370 | 5412 | 64‡ | 34 | 6,29 | **2,69** | 26 | 26 |
| cre-d | 4094 | 28601 | 281 | 131 | 27,76 | **25,88** | 42 | 42 |
| ex05 | 832 | 7805 | 96‡ | 61 | 12,37 | **5,01** | 65 | 39 |
| ex09 | 1821 | 18184 | 319 | 86 | 47,94 | **43,98** | 45 | 45 |
| osa-14 | 2300 | 54760 | 0 | 4 | – | **1,28** | – | 18 |
| osa-30 | 4313 | 104337 | 0 | 7 | – | **3,82** | – | 23 |
| osa-60 | 10243 | 243209 | 0 | 6 | – | **14,54** | – | 23 |
| ken11 | 9964 | 16740 | 74 | 20 | 10,42 | **8,19** | 23 | 22 |
| ken13 | 22365 | 36561 | 73 | 38 | 94,63 | **56,10** | 29 | 29 |
| ken18 | 78538 | 128434 | 103 | 45 | 1052,51 | **901,08** | 41 | 37 |
| pds-06 | 9145 | 28472 | 216 | 60 | 8,34 | **7,88** | 39 | 39 |
| pds-10 | 16558 | 48763 | 256 | 69 | 18,49 | **16,78** | 47 | 47 |
| pds-20 | 32276 | 106180 | 322 | 90 | 212,79 | **210,62** | 60 | 61 |
| pds-40 | 34265 | 214385 | 479 | 135 | 410,47 | **396,32** | 78 | 79 |
| pds-60 | 96503 | 332862 | 492 | 150 | 1096,77 | **1064,20** | 84 | 84 |
| pds-80 | 126109 | 430800 | 478 | 166 | **1526,79** | 1597,84 | 83 | 83 |
| pds-100 | 156243 | 514577 | 508 | 190 | 2631,49 | **2448,85** | 87 | 87 |

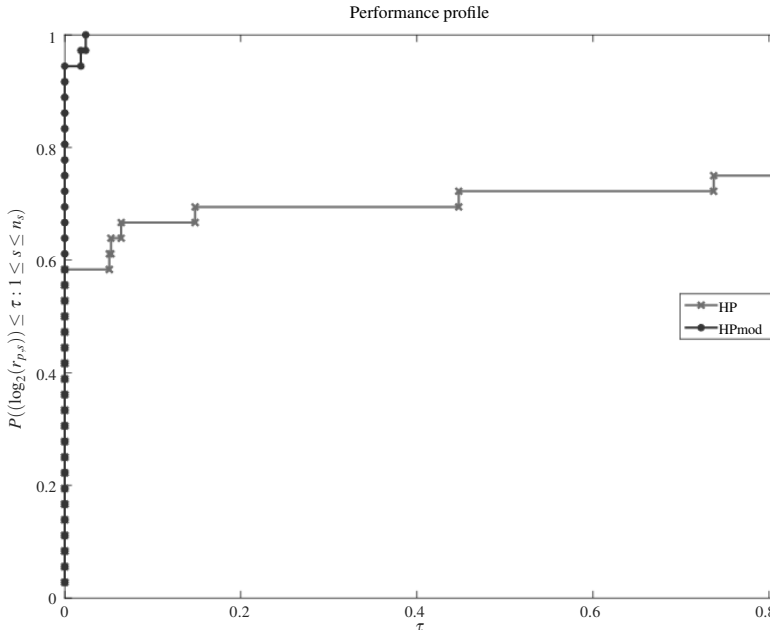


Figure 1: Performance profile for iterations of IPM.

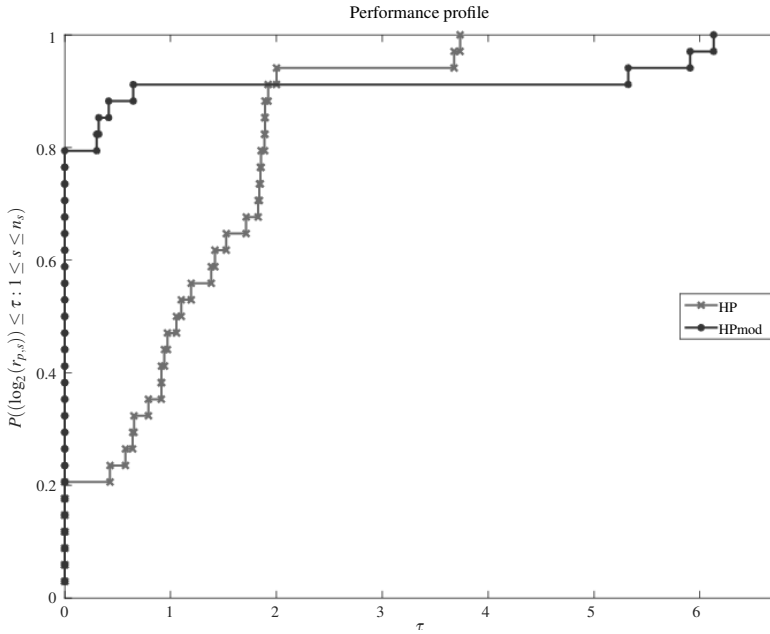


Figure 2: Performance profile for restars to compute CCF preconditioner.

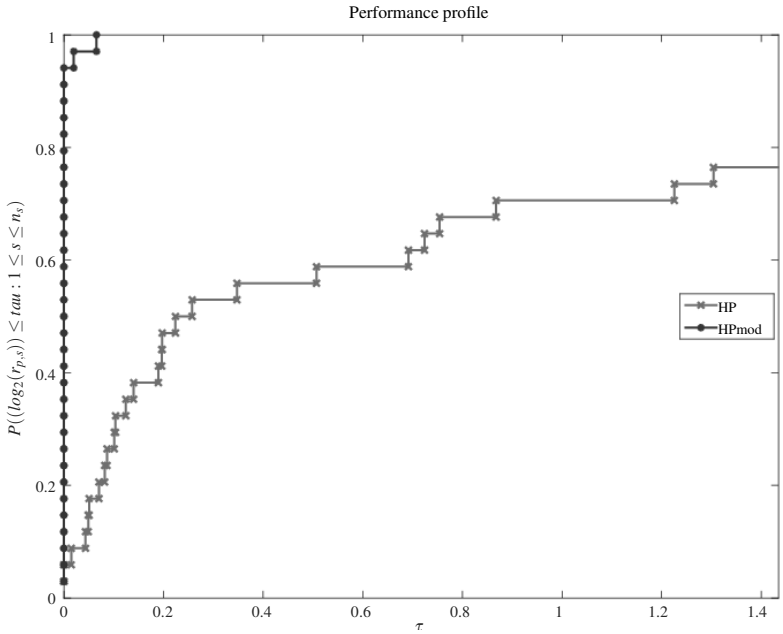


Figure 3: Performance profile for time of IPM.

early it may result in an increase in time or even in not finding the optimal solution. It can be seen in Fig. 3 that the HPmod performed better than the other approaches in 32 problems. HPmod solves all the problems, since the curve of its performance profile has reached 1.

It is observed that in 14 problems the number of IPM iterations was reduced when we used the HPmod. Problems qap12, osa-14, osa-30, osa-60, nug05-3rd, nug06-3rd, nug07-3rd, and nug08-3rd were not solved by HP. With respect to the time used to solve the problems, HPmod was superior in 32 of the 34 problems presented. Finally, in the fifth column of Table 1, it can be seen that in the HPmod approach, less than 15 restarts were performed to compute the CCF preconditioner in all IPM iterations.

## 6 CONCLUSIONS

The modifications in the CCF preconditioner in both the diagonal fault correction parameter $\alpha$ and the fill-in parameter $\eta$ reduced the number of restarts in the computation of this preconditioner. With the diagonal fault correction parameter of the original CCF preconditioner, see (3.6), up to fifteen attempts to build this preconditioner were allowed in more than one IPM iteration. With the new proposal, that did not happen in any IPM iteration. It resulted in the number of the preconditioned CGM iterations was reduced and therefore the processing time corresponding to the first phase of the HPmod also decreased.

In the SP, the computation of the base was accelerated because the sparse columns generated less fill and in addition, the proposed ordering for the SP performed well because the number of iterations of the preconditioned CGM did not increase and the effort to compute $B$ decreased.

## ACKNOWLEDGMENT

We would like to thank the agencies CNPq and FAPESP for grants which supported this research.

**RESUMO.** Este trabalho visa melhorar o cálculo da direção de busca no Método de Pontos Interiores primal-dual usando métodos iterativos precondicionados. Trata-se de uma abordagem híbrida que combina o precondicionador Fatoração Controlada de Cholesky e o precondicionador Separador. Esta abordagem tem mostrado bons resultados, entretanto, nesses précondicioandores existem fatores que reduzem sua eficiência, como falhas na diagonal ao calcular a Fatoração Incompleta de Cholesky, assim como a demanda por memória excessiva no precondicionador Separador, entre outros. Assim, algumas modificações são propostas nestes precondicionadores, bem como uma nova mudança de fase, a fim de melhorar o desempenho da aboradagem híbrida. Na Fatoração Controlada de Cholesky, os parâmetros que controlam o preenchimento e a correção das falhas que ocorrem na diagonal são modificados, para tal considera-se a relação entre os componentes da Fatoração Controlada de Cholesky obtidos antes e depois da falha na diagonal. No precondicionador Separador, por sua vez, a base esparsa é construída usando um ordenamento apropriado das colunas da matriz do problema de otimização. Além disso, é apresentado um resultado teórico que mostra que, com a ordenação proposta, o número da condição da matriz de Equações Normais precondicionada com o Precondicionador Separador é uniformemente limitado por uma quantidade que depende apenas dos dados originais do problema e não da iteração do Método do Pontos Interiores. Experimentos numéricos com problemas de grande porte corroboram a robustez e eficiência computacional desta abordagem.

**Palavras-chave:** Método de Ponto Interior, Fatoração Controlada de Cholesky, precondicionador Separador.

## REFERENCES

[1] G. Al-Jeiroudi, J. Gondzio & J. Hall. Preconditioning indefinite systems in interior point methods for large scale linear optimisation. *Optimization Methods and Software*, **23**(3) (2008), 345–363. doi:10.1080/10556780701535910. URL `https://doi.org/10.1080/10556780701535910`.

[2] S. Bellavia, V. Simone, D. Serafino & B. Morini. A preconditioning framework for sequences of diagonally modified linear systems arising in optimization. *SIAM Journal on Numerical Analysis*, **50**(6) (2012), 3280–3302.

[3] S. Bocanegra, F.F. Campos & A.R.L. Oliveira. Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods. *Computational Optimization and Applications*, **36**(2-3) (2007), 149–164.

[4] F.F. Campos. "Analysis of conjugate gradients-type methods for solving linear equations.". Ph.D. thesis, University of Oxford (1995).

[5] L. Casacio, C. Lyra, A.R.L. Oliveira & C.O. Castro. Improving the Preconditioning of Linear Systems from Interior Point Methods. *Comput. Oper. Res.*, **85**(C) (2017), 129–138. doi:10.1016/j.cor.2017.04.005. URL https://doi.org/10.1016/j.cor.2017.04.005.

[6] J. Czyzyk, S. Mehrotra, M. Wagner & S.J. Wright. PCx An Interior Point Code for Linear Programming. *Optimization Methods & Software*, **11** (1999), 397–430.

[7] E.D. Dolan & J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, **91**(2) (2002), 201–213.

[8] J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, **6**(2) (1996), 137–156. doi:10.1007/BF00249643. URL https://doi.org/10.1007/BF00249643.

[9] J. Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, **218**(3) (2012), 587–601.

[10] J. Gondzio. Matrix-Free Interior Point Method. *Computacional Optimization and Applications*, **51** (2012), 457–480.

[11] M.R. Heredia & A.R.L. Oliveira. Uma nova proposta para modificar a Fatoração Controlada de Cholesky no método dos pontos interiores. **1** (2015), 2912–2923.

[12] M.T. Jones & P.E. Plassmann. An improved incomplete Cholesky factorization. *ACM Transactions on Mathematical Software (TOMS)*, **21**(1) (1995), 5–17.

[13] C.J. Lin & J.J. Moré. Incomplete Cholesky factorizations with limited memory. *SIAM Journal on Scientific Computing*, **21**(1) (1999), 24–45.

[14] I.J. Lustig, R.E. Marsten & D.F. Shanno. On implementing Mehrotras s predictor–corrector interior-point method for linear programming. *SIAM Journal on Optimization*, **2**(3) (1992), 435–449.

[15] T.A. Maunteuffel. An incomplete factorization technique for positive definite linear systems. *Mathematics of computation*, **34**(150) (1980), 473–497.

[16] R.D. Monteiro, J.W. O'Neal & T. Tsuchiya. Uniform boundedness of a preconditioned normal matrix used in interior-point methods. *SIAM Journal on Optimization*, **15**(1) (2004), 96–100.

[17] A.R.L. Oliveira & D. Sorensen. A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. *Linear Algebra and its applications*, **394** (2005), 1–24.

[18] L.M. Silva & A.R.L. Oliveira. Melhoria do desempenho da fatoração controlada de Cholesky no precondicionamento de sistemas lineares oriundos dos métodos de pontos interiores. In "Proceeding Series of the Brazilian Society of Computational and Applied Mathematics", volume 3. SBMAC (2015), pp. 1–7.

[19] P. Suñagua & A.R.L. Oliveira. A new approach for finding a basis for the splitting preconditioner for linear systems from interior point methods. *Computational Optimization and Applications*, **67**(1) (2017), 111–127. URL https://EconPapers.repec.org/RePEc:spr:coopap:v:67:y:2017:i:1:d:10.1007_s10589-016-9887-0.

[20] M.I. Velazco, A.R.L. Oliveira & F.F. Campos. A note on hybrid preconditioners for large-scale normal equations arising from interior-point methods. *Optimization Methods Software*, **25**(2) (2010), 321–332. doi:10.1080/10556780902992829.

[21] S.J. Wrigth. "Primal-dual Interior-Point Methods:". SIAM e-books. Society for Industrial and Applied Mathematics (SIAM) (1997).