Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: https://www.tandfonline.com/loi/goms20

Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization

Anna Altman & Jacek Gondzio

To cite this article: Anna Altman & Jacek Gondzio (1999) Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization, Optimization Methods and Software, 11:1-4, 275-302, DOI: 10.1080/10556789908805754

To link to this article: https://doi.org/10.1080/10556789908805754

Published online: 30 Jan 2008.

Optimization Meth. & Soft., 1999, Vol. 11&12, pp. 275–302
Reprints available directly from the publisher
Photocopying permitted by license only

REGULARIZED SYMMETRIC INDEFINITE SYSTEMS IN INTERIOR POINT METHODS FOR LINEAR AND QUADRATIC OPTIMIZATION*

ANNA ALTMAN[a,†] and JACEK GONDZIO[b,‡]

[a]*Systems Research Institute, Polish Academy of Sciences, Newelska 6, 01-447 Warsaw, Poland;* [b]*Department of Mathematics and Statistics, The University of Edinburgh, King's Buildings, Edinburgh, EH9 3JZ, Scotland*

(Received 27 March 1998; Revised 12 Febraury 1999; In final form 08 March 1999)

This paper presents linear algebra techniques used in the implementation of an interior point method for solving linear programs and convex quadratic programs with linear constraints. New regularization techniques for Newton systems applicable to both symmetric positive definite and symmetric indefinite systems are described. They transform the latter to quasidefinite systems known to be strongly factorizable to a form of Cholesky-like factorization.Two different regularization techniques, *primal* and *dual*, are very well suited to the (infeasible) primal-dual interior point algorithm. This particular algorithm, with an extension of multiple centrality correctors, is implemented in our solver HOPDM. Computational results are given to illustrate the potential advantages of the approach when applied to the solution of very large linear and convex quadratic programs.

Keywords: Linear programming; convex quadratic programming; symmetric quasidefinite systems; primal-dual regularization; primal-dual interior point method; multiple centrality correctors

1 INTRODUCTION

We are concerned in this paper with the solution of large scale linear and convex quadratic programs using the infeasible primal-dual interior

* The research of the first author was supported by the Polish Academy of Sciences. The research of the second author was partly supported by the Fonds National de la Recherche Scientifique Suisse, grant #12-42503.94 during his visit at Logilab, University of Geneva, 102 Bd Carl-Vogt, 1211 Geneva, Switzerland (on leave from the Systems Research Institute, Polish Academy of Sciences).

† Corresponding author.

‡ E-mail: gondzio@maths.ed.ac.uk

point method. More specifically, we are concerned with the linear algebra techniques applied within this algorithm.

The first theoretical result for the primal-dual method for linear programming is due to Megiddo [19] who proposed to apply a logarithmic barrier method to the primal and the dual problems at the same time. Independently, Kojima, Mizuno and Yoshise [15] developed the theoretical background of this method, and gave the first complexity results. Continuous improvements to the implementation of this method have led to the development of several very efficient LP codes [16,20]. The major progress in the implementation has been done by Lustig, Marsten and Shanno and was summarized in [17]. A review of today's computational practice of the primal-dual implementation is given by [4,30].

The primal-dual method has an obvious extension to convex quadratic optimization problems, see e.g., Vanderbei and Carpenter [29]. Its key implementational issue is the solution of the (presumably large and sparse) symmetric indefinite system arising in the Newton system

$$\begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} f \\ h \end{bmatrix}, \tag{1}$$

where $Q \in \mathcal{R}^{n \times n}$ is a symmetric positive definite matrix, $\Theta \in \mathcal{R}^{n \times n}$ is a diagonal scaling matrix (with strictly positive elements) and $A \in \mathcal{R}^{m \times n}$ is the matrix of linear constraints; we assume it has a full row rank. At every iteration of the primal-dual method such a system of equations has to be solved with one, two or more different right hand side vectors (f, h). This is usually done in two steps. In the first one, the factorization of the matrix into an easily invertible form is computed; this factorization is then used as many times as is necessary in the solution of systems like (1).

The matrix in (1) is symmetric but not positive definite, so one cannot apply the Cholesky factorization to it. Instead, the symmetry preserving Bunch-Parlett triangular factorization can be used

$$H = \begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} = LDL^T, \tag{2}$$

where L is a triangular matrix and D is a block diagonal matrix with 1×1 and 2×2 pivots.

Vanderbei [28] analysed closely related symmetric indefinite matrices of the form

$$H_V = \begin{bmatrix} -E & A^T \\ A & F \end{bmatrix}, \tag{3}$$

where $E \in \mathcal{R}^{n\times n}$ and $F \in \mathcal{R}^{m\times m}$ are symmetric positive definite matrices and $A \in \mathcal{R}^{m\times n}$ has a full row rank. He proved that a Cholesky-like factorization exists for any symmetric permutation of such a *quasidefinite matrix*

$$PH_VP^T = \tilde{L}\tilde{D}\tilde{L}^T, \tag{4}$$

with a *diagonal* matrix $\tilde{D}$. This diagonal matrix has n strictly negative and m strictly positive elements (pivots).

In our method we choose pivots only from diagonal elements of the symmetric indefinite matrix. Consequently, the analysis phase is done prior to numerical factorization, just like in the case of factorizing a symmetric positive definite matrix. To get better accuracy of the factorization, Vanderbei [28] used a special ordering in which variables were separated into priority groups and ordered within these groups. For example, the pivoting on variables corresponding to free columns or dense columns was delayed. Additionally, to get from the standard system (1), with indefinite matrix H, to a quasidefinite strongly factorizable matrix H_V, he added slack variables to all linear constraints including equality type rows.

In this paper we propose to use a regularization technique to transform matrix H to a quasidefinite matrix. We want to avoid *any* reordering in the numerical phase of the factorization. To achieve this, we propose to use primal-dual regularization of the form

$$M = \begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} + \begin{bmatrix} -R_p & 0 \\ 0 & R_d \end{bmatrix}, \tag{5}$$

where diagonal matrices $R_p \in \mathcal{R}^{n\times n}$ and $R_d \in \mathcal{R}^{m\times m}$ can be interpreted as adding proximal terms to the primal and dual objective functions, respectively. We refer the reader to Rockafellar [22,23] for a discussion of the proximal point algorithm for convex programming and to Setiono [26] for its application to an interior point method.

A similar regularization was used by Saunders and Tomlin [24,25] for the same reason of allowing any ordering for the indefinite Cholesky factorization (4). However, to ensure stability *a priori*, they fixed the regularization terms in advance

$$R_p = \gamma^2 I_n \quad \text{and} \quad R_d = \delta^2 I_m, \tag{6}$$

with some predetermined regularization parameters γ and δ. We consider this to be an unnecessary restriction. Instead, we propose to adjust the regularizations dynamically. Namely, we use a negligibly small regularization

for all acceptable pivots and increase it whenever a dangerously small pivot candidate appears.

The use of *dynamic* regularization introduces less perturbation to the original logarithmic barrier method: the regularization concentrates uniquely on potentially unstable pivots.

This regularization technique can be incorporated easily into the standard Cholesky factorization. We modified the code of [12] accordingly. Moreover, it can be applied with both of the competitive approaches used to solve Newton equation systems: with the normal equations and with the augmented system. In the former approach, the primal regularization R_p has to be determined in advance, i.e., before the normal equations matrix

$$A(Q + \Theta^{-1} + R_p)^{-1}A^T \tag{7}$$

is formed and the dual regularization R_d is set up while factorizing this matrix. In the augmented system approach, both regularizations are chosen dynamically as the Cholesky-like factorization proceeds.

The paper is organized as follows. In Section 2 we recall basic ideas of the primal-dual method for convex quadratic problems with linear constraints. Additionally, we propose an extension of the multiple centrality correctors technique of Gondzio [14] to quadratic programming. In Section 3 we address the issues of linear algebra of a single interior point iteration and recall certain properties of the quasidefinite matrices. In Section 4 we show how to transform indefinite systems into quasidefinite ones. Then, in Section 5, we introduce the regularization technique and interpret it in terms of the primal-dual method. In Section 6 we present computational results which show the overall efficiency and accuracy of our HOPDM code when applied to solve linear and quadratic optimization problems. We illustrate the use of regularizations as well as the advantages resulting from the application of multiple centrality correctors. Finally, in Section 7, we present our conclusions.

2 PRIMAL-DUAL METHOD FOR QUADRATIC PROGRAMS

We would like to solve the general convex quadratic problem with linear constraints, i.e.

$$\text{minimize } c^T x + \frac{1}{2}x^T Q x,$$

$$\begin{aligned} &\text{subject to } Ax = b, \\ &\qquad x + s = u, \\ &\qquad x, \quad s \geq 0, \end{aligned} \tag{8}$$

where $c, x, s, u \in \mathcal{R}^n$, $b \in \mathcal{R}^m$, $A \in \mathcal{R}^{m \times n}$ is presumed to have full row rank. The matrix $Q \in \mathcal{R}^{n \times n}$ is positive semi-definite. The dual of (8) is given by

$$\begin{aligned} &\text{maximize } b^T y - u^T w - \frac{1}{2} x^T Q x, \\ &\text{subject to } A^T y + z - w - Qx = c, \\ &\qquad x, \quad z, \quad w \geq 0, \end{aligned} \tag{9}$$

where $z, w \in \mathcal{R}^n$, $y \in \mathcal{R}^m$.

For ease of presentation, all primal variables are assumed to be bounded in the above formulation. However, many linear and quadratic problems have *free* variables. Such variables may be replaced with a difference of two nonnegative variables, but this technique is known to introduce instability into computations. The use of primal regularization gives a natural way of handling free variables. Later in this section, we address this issue in more detail.

Applying the logarithmic barrier method [9], we augment the problems (8) and (9) by subtracting (adding) a logarithmic barrier term to the objective, which yields the following *primal barrier* and *dual barrier* problems:

$$\begin{aligned} &\text{minimize } c^T x + \frac{1}{2} x^T Q x - \mu \sum_{j=1}^{n} (\ln x_j + \ln s_j), \\ &\text{subject to } Ax = b, \\ &\qquad x + s = u, \\ &\qquad x, \quad s > 0 \end{aligned} \tag{10}$$

and

$$\begin{aligned} &\text{maximize } b^T y - u^T w - \frac{1}{2} x^T Q x + \mu \sum_{j=1}^{n} (\ln z_j + \ln w_j), \\ &\text{subject to } A^T y + z - w - Qx = c, \\ &\qquad x \geq 0, \quad z, \quad w > 0. \end{aligned} \tag{11}$$

The first order optimality conditions for (10) and (11) are

$$\begin{aligned} Ax &= b, \\ x + s &= u, \\ A^T y + z - w - Qx &= c, \\ XZe &= \mu e, \\ SWe &= \mu e, \\ x, \quad s, \quad z, \quad w &> 0, \end{aligned} \tag{12}$$

where X, S, Z, W are diagonal matrices built from vectors x, s, z, w, respectively, e is the vector of all ones and μ is a strictly positive barrier parameter.

Similarly to the linear programming case, primal-dual method for quadratic programming proceeds in the following way. For a given μ, the system (12) is built and one step of the damped Newton method towards its solution is executed. Then μ is updated (usually by decreasing). The algorithm continues until both duality gap and infeasibilities drop below the required tolerances.

The Newton equation system corresponding to a fixed barrier parameter μ has the form

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ I & I & 0 & 0 & 0 \\ -Q & 0 & A^T & I & -I \\ Z & 0 & 0 & X & 0 \\ 0 & W & 0 & 0 & S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta y \\ \Delta z \\ \Delta w \end{bmatrix} = \begin{bmatrix} \xi_b \\ \xi_u \\ \xi_c \\ \xi_z \\ \xi_w \end{bmatrix}, \tag{13}$$

where

$$\xi_b = b - Ax, \tag{14}$$

$$\xi_u = u - x - s, \tag{15}$$

$$\xi_c = c - A^T y - z + w + Qx, \tag{16}$$

$$\xi_z = \mu e - Xz, \tag{17}$$

$$\xi_w = \mu e - Sw, \tag{18}$$

are the violations of the first order optimality conditions. Its solution, $(\Delta x, \Delta s, \Delta y, \Delta z, \Delta w)$, defines the primal-dual search direction.

The above system can be transformed into

$$\begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} f \\ h \end{bmatrix}, \tag{19}$$

where

$$\Theta^{-1} = X^{-1}Z + S^{-1}W, \tag{20}$$

$$f = \xi_c - X^{-1}\xi_z + S^{-1}\xi_w - S^{-1}W\xi_u, \tag{21}$$

$$h = \xi_b. \tag{22}$$

Note that there are two essential differences between the primal-dual method for linear and for quadratic programming. They result from the presence of a quadratic term in the objective. In the algorithm for quadratic programming:

- Q appears in the system (19);
- Qx appears in the dual constraints and in the right hand side of the Newton equation system (16).

In Section 3 we shall address these issues in more detail.

Let us now add two comments concerned with the formulation of quadratic program (8).

If the problem contains a free variable x_j, then there is no slack variable s_j associated with it, and the dual slack z_j is equal to zero. This means that the jth element in matrix Θ^{-1} in (19) is set to zero. In such a case, for semidefinite Q there would be no guarantee that the matrix in (19) is nonsingular. Let us observe that the sole use of dual regularization could not guarantee the nonsingularity of this matrix. In contrast, the use of primal regularization is a natural way to get a nonsingular matrix.

There exists a special class of so-called *separable* quadratic programming problems in which matrix Q is diagonal. Solving separable quadratic problems does not differ much from solving linear problems since the upper left block in (19) remains diagonal (as in the linear programming case). Certain quadratic programming problems are formulated in such a way that a symmetric matrix Q is obtained as a product of F^TF, where $F \in \mathcal{R}^{p\times n}$. If matrix F is known *a priori*, then the original quadratic program can be reformulated to a simpler linearly constrained quadratic program with a diagonal quadratic term in the objective. Altman [1,2] extended the LP interior point code HOPDM [13] to this class of convex

quadratic problems and applied it in a number of real-life applications, see e.g. [3].

In this paper, we are concerned with the general convex quadratic optimization problems.

For our discussion in this section, it is sufficient to note that the solution of the system (19) is the main computational effort in a single iteration of the primal-dual method for quadratic programming. Sometimes the factorization of the system (19) is very expensive. It is then computationally advantageous to exploit the factorization further by using it in a sequence of solves of system (19) with different right hand sides. This is the approach used in our implementation: we use Mehrotra's predictor-corrector technique [20], implemented in a similar way as in the linear programming case [16], and Gondzio's multiple centrality correctors [14]. The generalization of these correcting techniques from linear to quadratic programming is quite straightforward; hence we shall address it only very briefly.

2.1 Predictor-corrector Technique

Mehrotra's predictor-corrector strategy [20,16] has two components: an adaptive choice of the barrier parameter and the computation of a high-order approximation to the central path.

The first step of the predictor-corrector strategy is to compute the affine scaling (predictor) direction. The affine scaling direction solves the Newton equation system (13) with $\mu = 0$ and is denoted by Δ_a.

After the affine scaling direction has been computed, the maximum stepsizes along this direction in the primal (α_{Pa}) and in the dual (α_{Da}) spaces are determined preserving nonnegativity of (x, s) and (z, w). Next, the predicted complementarity gap

$$g_a = (x + \alpha_{Pa}\Delta x)^T(z + \alpha_{Da}\Delta z) + (s + \alpha_{Pa}\Delta s)^T(w + \alpha_{Da}\Delta w)$$

is computed and the barrier parameter is chosen using the heuristic

$$\mu = \left(\frac{g_a}{g}\right)^2 \frac{g_a}{n}. \tag{23}$$

Next, the second (corrector) component of the predictor-corrector direction is computed. Note that we ideally want to compute a direction such

that the next iterate is perfectly centered, i.e.

$$(X + \Delta X)(z + \Delta z) = \mu e. \tag{24}$$

(We have a similar relation for variables s and w associated with the upper bounds). The above system can be rewritten as

$$Z\Delta x + X\Delta z = -Xz + \mu e - \Delta X \Delta z. \tag{25}$$

Let us observe that in the computations of the Newton direction in equation (13), the second order term $\Delta X \Delta z$ is neglected. Instead of setting the second order term equal to zero, Mehrotra proposes estimating $\Delta X \Delta z$ by the affine-scaling direction $\Delta X_a \Delta z_a$. His predictor-corrector direction is obtained by solving the Newton equations system, with (25) as the linearized complementarity conditions and the barrier parameter μ chosen by (23).

A single iteration of the predictor-corrector primal-dual method needs two solves of the same large, sparse linear system for two different right hand sides. Computational practice shows that the additional cost of the predictor-corrector strategy is more than offset by a reduction in the number of iterations (factorizations).

Many large scale quadratic programs exist for which the factorizations are extremely expensive. For those problems, reducing the number of factorizations becomes more important. Unfortunately, as was shown by Carpenter *et al.* [8], the repeated use of Mehrotra's corrector does not offer computational advantages.

2.2 Modified Centering Directions

Let us observe that the step $(\Delta x, \Delta y, \Delta s, \Delta z, \Delta w)$ of (13) aims to draw all complementarity products to the same value μ. Moreover, to ensure the progress of optimization, the barrier parameter μ has to be smaller than the average complementarity product $\mu_{average} = (x^T z + s^T w)/2n$. Such perfectly centered points usually cannot be reached.

The approach proposed by Gondzio for linear programming [14] applies multiple centrality corrections and combines their use with a choice of reasonable, well centered *targets* that are supposed to be easier to reach than perfectly centered (but usually unreachable) analytic centers. The method generalizes easily to quadratic programming.

Assume (x, s) and (y, z, w) are primal and dual solutions at a given iteration of the primal-dual algorithm (x, s, z and w are strictly positive). Next, assume that a *predictor* direction Δ_p at this point is determined and the maximum stepsizes in primal, α_P and dual, α_D spaces are computed that preserve nonnegativity of the primal and dual variables, respectively.

We look for a *corrector* direction Δ_m such that larger stepsizes in the primal and dual spaces are allowed for a composite direction

$$\Delta = \Delta_p + \Delta_m. \tag{26}$$

To enlarge these stepsizes from α_P and α_D to $\tilde{\alpha}_P = \min(\alpha_P + \delta_\alpha, 1)$ and $\tilde{\alpha}_D = \min(\alpha_D + \delta_\alpha, 1)$, respectively, a corrector term Δ_m has to compensate for the negative components in the primal and dual variables

$$\begin{aligned}(\tilde{x}, \tilde{s}) &= (x, s) + \tilde{\alpha}_P(\Delta_p x, \Delta_p s),\\ (\tilde{y}, \tilde{z}, \tilde{w}) &= (y, z, w) + \tilde{\alpha}_D(\Delta_p y, \Delta_p z, \Delta_p w).\end{aligned} \tag{27}$$

We try to reach this goal by adding the corrector term Δ_m that drives from this exterior trial point to the next iterate $(\hat{x}, \hat{s}, \hat{y}, \hat{z}, \hat{w})$ lying in the vicinity of the central path. However, we are aware that there is little chance to reach the analytic center in one step, that is to reach $v = (\mu e, \mu e) \in \mathcal{R}^{2n}$ in the space of the complementarity products. Hence, we compute the complementarity products of the trial point $\tilde{v} = (\tilde{X}\tilde{z}, \tilde{S}\tilde{w}) \in \mathcal{R}^{2n}$, and concentrate the effort on correcting only their outliers. We thus project the point $\tilde{v}$ componentwise on a hypercube $H = [\beta_{min}\mu, \beta_{max}\mu]^{2n}$ to get the following target

$$v_t = \pi(\tilde{v}|H) \in \mathcal{R}^{2n}. \tag{28}$$

The corrector direction Δ_m solves the linear system similar to (13) for the following right hand side

$$(0, 0, 0, v_t - \tilde{v}) \in \mathcal{R}^{4n+m}, \tag{29}$$

with nonzero elements only in a subset of positions of $v_t - \tilde{v}$ that refer to the complementarity products which do not belong to $(\beta_{min}\mu, \beta_{max}\mu)$.

Once the corrector term Δ_m is computed, the new stepsizes $\hat{\alpha}_P$ and $\hat{\alpha}_D$ are determined for the composite direction

$$\Delta = \Delta_p + \Delta_m, \tag{30}$$

and the primal-dual algorithm can move to the next iterate.

The correcting process can be repeated any number of times. In such a case, the direction Δ of (30) becomes a new predictor Δ_p and is used to compute the new trial point (27). An advantage of this approach is that computing every single corrector term needs exactly the same effort (it is dominated by the solution of the system like (13) with the right hand side (29)).

The number of centrality correctors to use depends on the ratio of the factorization effort and the backsolve effort involved when solving equations like (13). The larger this ratio, the (relatively) cheaper one additional backsolve becomes, so more correctors can be tried (see [14] for details).

3 LINEAR ALGEBRA IN INTERIOR POINT METHOD

We noted in the previous section that the large Newton equations system reduces to the so-called *augmented system* (19) that involves the symmetric indefinite matrix

$$H = \begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix}. \tag{31}$$

Whenever the solution of the $n \times n$ system with the positive definite matrix $Q + \Theta^{-1}$ is inexpensive, one could eliminate

$$\Delta x = (Q + \Theta^{-1})^{-1}(A^T \Delta y - f)$$

from (19) and obtain the normal equations system

$$A(Q + \Theta^{-1})^{-1} A^T \Delta y = A(Q + \Theta^{-1})^{-1} f + h. \tag{32}$$

This reduction is particularly attractive for systems arising in linear programming when $Q = 0$. Actually, such a reduction is done in many commercial and public domain interior point codes, see e.g., [4]. Its main advantage follows from the fact that the matrix involved in system (32) is positive definite, so the reordering for sparsity can be done before the optimization starts.

The reduction of H to the normal equations is nothing but the explicit elimination of block $Q + \Theta^{-1}$. This is equivalent to choosing n diagonal elements of $Q + \Theta^{-1}$ as the first n pivots while factorizing H. Clearly, one could expect that leaving more freedom in the pivot choice, i.e., factorizing directly the whole matrix H, could lead to a sparser (and cheaper)

factorization. Unfortunately, it is not possible, in general, to separate the reordering for sparsity from the numerical factorization of H since this matrix is not positive definite. Following the classical approach, one should use the Bunch-Parlett factorization [7,6] that chooses between 1×1 and (possibly indefinite) 2×2 pivots. The Bunch-Parlett factorization [10], like any other factorization that combines dynamic reordering with the numerical operations, is usually more expensive than its counterpart in which these operations are separated.

There exists a class of indefinite matrices for which these operations can be separated. Following [28], we describe this class below.

A symmetric matrix K is called *quasidefinite* if it has the form

$$K = \begin{bmatrix} -E & A^T \\ A & F \end{bmatrix}, \tag{33}$$

where $E \in \mathcal{R}^{n \times n}$ and $F \in \mathcal{R}^{m \times m}$ are positive definite and $A \in \mathcal{R}^{m \times n}$ has full row rank.

Symmetric nonsingular matrix K is *factorizable* if there exists a diagonal matrix D and unit lower triangular matrix L such that $K = LDL^T$. The symmetric matrix K is *strongly factorizable* if for any permutation matrix P a factorization $PKP^T = LDL^T$ exists. Vanderbei [28] proved that symmetric quasidefinite matrices are strongly factorizable.

4 FROM INDEFINITE TO QUASIDEFINITE SYSTEM

Unfortunately, the indefinite matrix H in (31) is not quasidefinite. To make it so, Vanderbei [27,28] added slack variables x_s to all linear constraints and added a new constraint forcing this variable to be zero at the optimum, $x_s + x_p = 0$. This operation requires the introduction of additional dual slack variables z_s and z_p. The idea of [27] is to reformulate the primal problem as follows:

$$\begin{aligned} \text{minimize } & c^T x + \frac{1}{2} x^T Q x, \\ \text{subject to } & Ax + x_s = b, \\ & x_s + x_p = 0, \\ & x, \quad x_s, \quad x_p \geq 0. \end{aligned} \tag{34}$$

The corresponding dual is:

$$\begin{aligned} &\text{maximize } b^T y - \frac{1}{2} x^T Q x, \\ &\text{subject to } A^T y + z - Qx = c, \\ &\qquad y - z_s + z_p = 0, \\ &\qquad x, \quad z, \quad z_s, \quad z_p \geq 0. \end{aligned} \tag{35}$$

The reduced Newton equation system for this primal-dual pair is written as

$$H_V = \begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & F \end{bmatrix}, \tag{36}$$

where $F = (Z_s X_s^{-1} + Z_p X_p^{-1})^{-1}$ is a positive definite matrix; hence H_V is quasidefinite. Let us observe that the primal slack variable x_s can be interpreted as a certain form of regularization.

Another approach, due to Saunders and Tomlin [25], consists of adding a perturbation to matrix H

$$H_{ST} = \begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} + \begin{bmatrix} -\gamma^2 I_n & 0 \\ 0 & \delta^2 I_m \end{bmatrix}. \tag{37}$$

Under the condition that the perturbation parameters γ and δ are not too small, namely

$$\gamma\delta \geq \sqrt{\varepsilon},$$

where ε is the relative machine precision, and $\|A\|$ is of order 1, the relative errors of the solution found by using H_{ST} can be bounded, cf. Saunders [24].

Let us note that Setiono [26] analysed an interior proximal point algorithm for LP in which a quadratic term with the matrix $\alpha^k I$ was added to the objective at every iteration. In our approach, the matrix $\alpha^k I$ is replaced by the matrix R_p with dynamically chosen entries; this gives us more flexibility in the choice of regularization.

5 REGULARIZATION IN THE PRIMAL AND DUAL SPACES

We propose a different approach that uses the following regularization

$$M = \begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} + \begin{bmatrix} -R_p & 0 \\ 0 & R_d \end{bmatrix}, \tag{38}$$

with R_p and R_d diagonal and chosen dynamically while computing Cholesky-like factorization of M. Let us start by giving an interpretation of the regularizations R_p and R_d.

Assume that x_0 is the current value of the primal variables x. Instead of formulating the primal barrier problem (10), let us consider a *primal regularized problem*:

$$\begin{aligned} \text{minimize } & c^T x + \frac{1}{2} x^T Q x + \frac{1}{2}(x - x_0)^T R_p (x - x_0) \\ & - \mu \sum_{j=1}^{n} (\ln x_j + \ln s_j), \\ \text{subject to } & Ax = b, \\ & x + s = u, \\ & x, \quad s > 0, \end{aligned} \tag{39}$$

with some nonnegative definite diagonal matrix R_p. The new quadratic term present in the objective, $(x - x_0)^T R_p (x - x_0)$, penalizes directions that move the current iterate far away from the *primal proximal point* x_0. The optimality conditions for this problem are

$$\begin{aligned} Ax &= b, \\ x + s &= u, \\ A^T y + z - w - Qx - R_p(x - x_0) &= c, \\ Xz &= \mu e, \\ Sw &= \mu e, \\ x, \quad s, \quad z, \quad w &> 0. \end{aligned} \tag{40}$$

The reduced Newton equation system is written

$$\begin{bmatrix} -Q - \Theta^{-1} - R_p & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} f' \\ h \end{bmatrix}, \tag{41}$$

where

$$f' = f - R_p(x - x_0)$$

replaces f in (19).

By analogy, assume that y_0 is the current value of the dual variables y and replace the dual barrier problem (11) with the following *dual regularized problem*:

$$\begin{aligned} \text{maximize } & b^T y - u^T w - \frac{1}{2} x^T Q x - \frac{1}{2}(y - y_0)^T R_d (y - y_0) \\ & + \mu \sum_{j=1}^{n} (\ln z_j + \ln w_j), \\ \text{subject to } & A^T y + z - w - Q x = c, \\ & x \geq 0, \quad z, \quad w > 0, \end{aligned} \tag{42}$$

where R_d is positive definite (diagonal) matrix. Again, the new quadratic term present in the objective, $(y - y_0)^T R_d (y - y_0)$, penalizes directions that move the current iterate far away from the *dual proximal point* y_0. The optimality conditions for this problem are

$$\begin{aligned} A x + R_d (y - y_0) &= b, \\ x + s &= u, \\ A^T y + z - w - Q x &= c, \\ X z &= \mu e, \\ S w &= \mu e, \\ x, \quad s, \quad z, \quad w &> 0. \end{aligned} \tag{43}$$

The corresponding Newton equation system reduces to

$$\begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & R_d \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} f \\ h' \end{bmatrix}, \tag{44}$$

where

$$h' = h - R_d (y - y_0)$$

replaces h in (19).

In our approach we use both regularizations at the same time. In other words, at every iteration of the interior point method we compute the

Newton direction for the following system of equations

$$\begin{aligned} Ax + R_d(y - y_0) &= b, \\ x + s &= u, \\ A^T y + z - w - Qx - R_p(x - x_0) &= c, \\ Xz &= \mu e, \\ Sw &= \mu e, \\ x, \quad s, \quad z, \quad w &> 0. \end{aligned}$$

This system differs from the optimality conditions (12) in added regularization terms R_p and R_d. The corresponding Newton equation system reduces to

$$\begin{bmatrix} -Q - \Theta^{-1} - R_p & A^T \\ A & R_d \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} f' \\ h' \end{bmatrix}.$$

We combine two regularizations to achieve stability in the linear algebra kernel. However, we keep the regularizations small since we do not want to change greatly the behavior of the primal-dual method for quadratic programming.

Let us observe that by the choice of (x_0, y_0) as the current primal-dual pair, we ensure that the right hand side of the reduced system (19) is not altered, i.e.,

$$f' = f \quad \text{and} \quad h' = h, \tag{45}$$

while we benefit from the presence of regularization

$$\begin{bmatrix} -R_p & 0 \\ 0 & R_d \end{bmatrix}$$

in the matrix (38).

The Newton direction $(\Delta x, \Delta y)$ corresponding to the regularized system would of course be slightly different from the one obtained from (19). We do not want to slow down the convergence of the primal-dual method by choosing uniform large regularizations like (37). We choose our regularizations dynamically. In the case of acceptable pivots, small terms are used; in the case of dangerously small pivot candidates, larger regularizations are used. Moreover, the choice of regularization is delayed until the pivot candidate is determined. This requires a simple modification of the Cholesky factorization that we have implemented within the code of [12].

The new routine which computes the generalized Cholesky factorization receives not only the original matrix H, permuted symmetrically to reduce fill-in, but also an additional integer vector indicating the type of pivot rows. Recall that negative pivots are expected in rows corresponding to $Q + \Theta^{-1}$ block and positive pivots are expected in rows corresponding to matrix A. The distinction between the pivots has to be maintained correctly if we do not want to loose the quasidefinite property of the matrix. The only modification of the factorization routine, compared with the Cholesky scheme, is an additional analysis of the pivot candidate. Once the pivot element d_{ii} is computed, we determine its type and check if its absolute value is too small. If this is the case, we replace the pivot with $d_{ii} - r_p^i$, if it is a "negative" type pivot, or with $d_{ii} + r_d^i$, if it is a "positive" type pivot. The regularizations R_p and R_d are stored separately, hence they are available outside the numerical factorization routine. They are needed when an iterative improvement procedure [5] is used to improve the accuracy of solutions.

Let us now explain what we understand by acceptable and dangerously small pivot candidates. We choose the maximum diagonal element h_{max} in the matrix H and consider all pivots d_{ii} such that $|d_{ii}| \geq \varepsilon h_{max}$, where ε is the relative machine precision, as acceptable. Pivot candidates with the absolute value smaller than εh_{max} will be corrected.

After extensive testing, we have found that the use of nonnegligible regularizations almost always slows down the primal-dual method (although it improves the accuracy of computations). Therefore we keep the default regularizations as small as possible. Moreover, we choose the default primal regularization to be smaller than the dual one and set

$$r_p = \varepsilon^{3/4} \quad \text{and} \quad r_d = \varepsilon^{1/2}. \tag{46}$$

These settings are used as long as a residual

$$r = \begin{bmatrix} f \\ h \end{bmatrix} - M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{47}$$

satisfies $\|r\|_\infty \leq \varepsilon^{1/2}$. If the residual in solution $(\Delta x, \Delta y)$ exceeds $\varepsilon^{1/2}$, we enable the iterative refinement procedure for the augmented system with the matrix M. For more than 80% of real-life problems, the iterative refinement routine is not activated except for the last interior point iteration. It is also very rare that more than one iterative refinement step has to be done to get a sufficiently accurate solution. Should this happen,

the second (and the last) step of the iterative refinement is executed and in the following interior point iteration the default regularizations (46) are multiplied by 10.

6 COMPUTATIONAL RESULTS

We shall discuss in this section the computational results that illustrate the accuracy and efficiency of the HOPDM code. The latest version uses the regularization technique described in the previous section. We apply our program to the solution of large scale linear and quadratic optimization problems.

All test problems used in our computational experiments come from public domain sources. Deterministic equivalents of stochastic optimization problems can be retrieved from:

`ftp://ftp.sztaki.hu/pub/oplab/LPTESTSET/STOCHLP/.`

The remaining linear optimization examples come from Netlib [11] collection:

`ftp://ftp.netlib.org/lp/data/.`

Quadratic programs [18] corresponding to Netlib's linear programs can be retrieved from:

`ftp://ftp.brunel.ac.uk/maths/qplib/qdata/.`

Finally, large scale quadratic programs, both the randomly generated ones and those coming from the CUTE collection, have been provided by Hans Mittelmann [21], see e.g.,

`http://plato.la.asu.edu/guide.html.`

Almost all our computations were done on a PC with a 200 MHz Pentium Pro processor and 128 MB of RAM run under Linux operating system. HOPDM was compiled with the FORTRAN GNU compiler `fort77`; options `-O2 -m486` were used. All solution times were measured with the Linux `time` command. Solution times are given in seconds and include reading the input file (MPS for LP and MPS-like for QP) and writing an MPS output report.

Large, sparse quadratic programs from the CUTE collection were solved on an IBM Power 2 workstation with a 58 MHz RS6000 processor and 128 MB of RAM run under AIX 4.2 operating system. HOPDM was compiled with the IBM FORTRAN compiler `xlf`; options `-O -qarch=pwr` were used. For these runs, pure solution times were measured (excluding reading the MPS input and writing solution reports).

In all tables the columns m and n show the number of rows and columns in constraint matrix A. nz(A) denotes the number of nonzeros in A. For other symmetric matrices — AA^T, Q and H in (31) — nz shows the number of nonzeros in the lower (or upper) triangle (without diagonal). The matrix L is the Cholesky (or Cholesky-like) factor of matrix AA^T or H.

6.1 Advantages of the Augmented System Factorization

Table I compares the efficiency of the normal equations approach with that of the augmented system. We ran HOPDM with two different factorization methods on one specific stochastic optimization problem, `scsd8-2r`, with the number of scenarios varying from 27 to 432. Stochastic linear problems contain dense columns and therefore solve more efficiently with the augmented system factorization. The data collected in Table I is self-explanatory: for both factorization techniques, we report the number of off-diagonal nonzero entries in the matrices being factorized; and in their triangular factors, as well as the optimal solution statistics (iterations and CPU time), all to 8 significant figures. From the analysis of results collected in Table I one can conclude that the augmented factorization very easily handles problems with dense columns. (This result is not new, of course.) We note that in the augmented system factorization, the number of nonzero elements in the factors L grows linearly with the number of scenarios.

TABLE I Factorization: normal equations vs. augmented system (Pentium Pro, 200 MHz).

Name	Densest column	Normal equations				Augmented system			
		$nz(AA^T)$	$nz(L)$	iters	time	$nz(H)$	$nz(L)$	iters	time
scsd8-2r-27	56	12435	30508	11	6.75	11794	25841	13	4.63
scsd8-2r-54	110	39435	110532	11	24.10	23458	45617	13	8.76
scsd8-2r-108	218	137175	419316	12	155.04	46786	89789	15	20.12
scsd8-2r-216	434	507615	1631712	13	1195.52	93442	178133	17	45.46
scsd8-2r-432	866	1948335	6435988	10	9426.95	186754	354821	16	91.53

TABLE II Solving stochastic optimization problems (Pentium Pro, 200 MHz).

Name	m	n	nz(A)	Dense	$nz(H)$	$nz(L)$	iters	time	IR	$\|r\|_\infty$
aircraft	3754	7517	24034	751	19667	19759	10	8.12	0	8e−13
deter3	7647	21777	44619	73	44474	84913	25	36.60	0	3e−09
deter7	6375	18153	37191	61	37070	70789	24	29.46	0	3e−09
fxm2.16	3900	5602	31412	36	28596	78439	25	18.50	0	6e−09
fxm3.16	41340	64162	372452	36	341076	725261	36	316.69	39	1e−06
fxm4.6	22400	30732	250142	23	238096	560766	29	161.02	5	5e−08
pltexpA2.16	1726	4540	11154	18	8289	20949	18	5.84	0	4e−10
pltexpA3.16	28350	74172	181650	18	135697	363991	42	212.07	0	5e−10
pltexpA4.6	26894	70364	172326	8	130315	1185774	34	609.95	0	2e−11
sc205.2r.800	17613	17614	48031	802	53638	100881	14	33.34	0	5e−11
sc205.2r.1600	35213	35214	96031	1602	107238	201681	23	112.42	13	3e−05
scagr7.2r.432	16431	17300	70477	1297	42372	75678	26	39.52	4	2e−08
scagr7.2r.864	32847	34580	140893	2593	84708	151278	28	91.13	8	3e−08
scfxm1.2r.128	19036	28914	107572	260	90228	277101	42	126.39	12	1e−06
scfxm1.2r.256	37980	57714	214452	516	179828	551533	42	260.16	3	4e−08
scrs8.2r.256	7196	9765	30035	257	8421	13044	13	9.06	0	2e−12
scrs8.2r.512	14364	19493	59987	513	17269	26864	15	20.94	0	1e−12
scsd8.2r.216	4330	30310	125480	434	93442	178133	17	45.03	0	3e−10
scsd8.2r.432	8650	60550	250760	866	186754	354821	16	90.24	0	4e−10
sctap1.2r.216	12990	20784	91728	650	81272	143210	21	48.63	0	1e−12
sctap1.2r.480	28830	46128	203664	1442	180536	317714	25	132.56	0	1e−13

In Table II we report results for the solution of several large stochastic linear programs. The column Dense shows the number of nonzeros in the densest column. Additionally, we report in this Table, the overall number of iterative refinement steps (IR) needed to improve the accuracy of the direction and the largest error $\|r\|_\infty$ in (47) throughout a given run of HOPDM. These results confirm the high accuracy of the regularized factorization.

6.2 From Linear to Quadratic Programs

There exists a collection of 46 convex (nonseparable) quadratic programs that are modified from linear programs from the Netlib set. The transformation consists in adding a nonseparable quadratic term Q to the objective of the original linear program. We solved both the linear problems and the corresponding quadratic problems and collected the results in Table III. We report the sizes of the linear program, the number of off-diagonal nonzero entries in the matrix Q of the quadratic program and the solution statistics (iterations and overall CPU time to get optimal solutions to 8 significant figures).

The analysis of results collected in Table III shows that quadratic programs are not necessarily more difficult to solve than linear programs,

TABLE III Solving linear and quadratic programs (Pentium Pro, 200 MHz).

Problem	Linear Program					Quadratic Program		
	m	n	nz(A)	iters	time	nz(Q)	iters	time
25fv47	821	1571	11127	22	7.85	59053	33	66.12
adlittle	56	97	465	12	0.17	70	10	0.18
afiro	27	32	88	7	0.05	3	10	0.07
bandm	305	472	2659	18	0.87	16	19	1.20
beaconfd	173	262	3476	11	0.53	9	18	1.05
bore3d	233	315	1525	15	0.42	50	14	0.63
brandy	220	249	2150	21	0.84	49	15	0.86
capri	271	353	1786	18	0.82	838	48	2.45
e226	223	282	2767	21	0.93	897	13	1.09
etamacro	400	688	2489	19	1.85	4069	44	18.41
fffff800	524	854	6235	26	2.89	1638	36	9.68
forplan	161	421	4916	19	1.08	546	37	2.53
gfrd-pnc	616	1092	3467	15	1.19	108	16	1.62
grow15	300	645	5665	14	1.62	462	13	1.89
grow22	440	946	8318	15	2.48	787	13	3.03
grow7	140	301	2633	14	0.72	327	12	0.86
israel	174	142	2358	21	0.81	656	16	0.83
pilotnov	975	2172	13129	15	9.75	391	26	20.90
recipe	91	180	752	9	0.20	30	16	0.32
sc205	205	203	552	11	0.32	10	10	0.35
scagr25	471	500	2029	16	0.85	100	15	0.92
scagr7	129	140	553	12	0.23	17	14	0.23
scfxm1	330	457	2612	17	0.94	677	31	2.10
scfxm2	660	914	5229	20	2.16	1057	35	5.13
scfxm3	990	1371	7846	21	3.46	1132	31	7.00
scorpion	388	358	1708	13	0.55	18	12	0.63
scrs8	490	1169	4029	19	1.56	88	16	1.77
scsd1	77	760	3148	9	0.63	691	9	0.82
scsd6	147	1350	5666	11	1.18	1308	11	1.66
scsd8	397	2750	11334	11	2.60	2370	11	3.91
sctap1	300	480	2052	15	0.72	117	16	0.95
sctap2	1090	1880	8124	16	3.33	636	13	3.74
sctap3	1480	2480	10734	16	4.47	861	13	5.56
seba	515	1028	4874	8	0.74	550	33	2.89
share1b	117	225	1182	19	0.48	21	16	0.53
share2b	96	79	730	13	0.28	45	25	0.48
shell	536	1775	4900	22	2.12	34385	22	7.20
ship04l	402	2118	8450	12	1.88	42	10	2.07
ship04s	402	1458	5810	12	1.37	42	11	1.56
ship08l	778	4283	17085	13	3.60	34025	11	29.94
ship08s	778	2387	9501	13	1.97	11139	12	6.40
ship12l	1151	5427	21597	14	4.98	60205	11	46.00
ship12s	1151	2763	10941	13	2.54	16361	13	8.24
sierra	1227	2036	9252	20	4.44	61	17	4.83
stair	356	467	3857	15	1.56	952	32	3.49
standata	359	1075	3038	15	1.16	666	12	1.42

for the primal-dual interior point method. Of course, solution times may increase, as can be seen, for example, with problems `25fv47` and `ship`. This is essentially a consequence of the need to factor a much denser matrix than in the linear programming case. Yet there is no considerable increase in the number of iterations needed to reach optimality for the quadratic program compared with that for the linear program.

6.3 Large Quadratic Programs

In the following two experiments we solved large and difficult quadratic optimization problems: relatively dense problems generated randomly by AMPL modeling language, and very large sparse programs from the CUTE collection of nonlinear problems. The results of running HOPDM on these test examples are collected in Tables IV and V, respectively. Except for problem CVXQP3, for which HOPDM stalled near 5 significant figures of the optimal solution, all problems were solved to the default 8 significant figures.

Our code behaves well on these problems. Moreover, the solution effort is predictable. Similarly to linear programs, quadratic problems also take advantage of the use of the multiple centrality correctors technique. It is worth noting that this technique limits the growth in the number of primal-dual iterations necessary to solve a given quadratic programming problem: this number grows sublinearly with the problem dimensions.

TABLE IV Solving randomly generated quadratic programs (Pentium Pro, 200 MHz).

Name	m	n	nz(A)	nz(Q)	nz(L)	iter	time
qp100-1	220	100	2542	1660	7555	12	1.20
qp100-2	2020	100	22206	1660	29056	12	5.96
qp100-3	220	100	22100	1660	27150	12	6.06
qp1000-1	700	1000	15753	68762	491059	14	339.09
qp250-1	550	250	14638	16771	46010	14	11.53
qp250-2	550	250	69140	30699	100515	12	30.31
qp250-3	550	250	14638	16771	46006	15	12.29
qp500-1	1100	500	56750	89071	181999	13	70.43
qp500-2	1100	500	51788	89071	177037	17	83.24
qp500-3	100	500	5614	89071	128837	7	33.52
sqp2500-1	2000	2500	52321	738051	3124093	14	5380.93
sqp2500-2	2000	2500	52319	14345	3504910	17	7363.16
sqp2500-3	4500	2500	115073	738051	3216994	16	6332.87

TABLE V Solving very large quadratic programs (IBM, Power 2, 58 MHz).

Name	m	n	nz	nz(Q)	nz(H)	nz(L)	iter	time
AUG2D	22500	45300	112500	0	112500	596270	8	66.77
AUG2DC	22500	45300	112500	0	112500	596270	9	75.02
AUG2DCQP	22500	45300	112500	0	112500	596270	10	73.06
AUG2DQP	22500	45300	112500	0	112500	596270	11	77.75
AUG3D	15625	52428	112981	0	112981	2851780	9	1405.07
AUG3DC	27000	89013	181320	0	181320	5641694	17	3628.78
AUG3DCQP	27000	89013	181320	0	181320	5641694	16	3310.65
AUG3DQP	15625	52428	112981	0	112981	2851780	13	1697.32
CVXQP1	7500	15000	22497	44981	67478	6619618	10	4186.08
CVXQP2	3750	15000	11249	44981	56230	5225664	9	3043.68
CVXQP3	11250	15000	33745	44981	78726	7872011	31*	15421.70
POWELL20	75000	75000	225000	0	225000	374997	7	58.05
UBH1	48000	72009	203982	0	203982	419664	15	80.86
YAO	60000	60002	239997	0	239997	499878	8	61.70

Consequently, quadratic problems with sizes reaching 100,000 variables can usually be solved in less than 20 interior point iterations.

Finally, we show the advantages of using multiple centrality correctors on three large problems AUG2DC, AUG3DC and CVXQP1. The Cholesky factors for these problems contain 596270, 5641694, and 6619618 subdiagonal entries, respectively. To see how the technique works, we let the maximum number of correctors go from 0 up to 10. By the heuristic of [14], the maximum number of centrality correctors allowed for these problems is set to 2, 10 and 10, respectively (the number of iterations in the runs corresponding to these default choices are marked in bold).

An analysis of the results collected in Table VI clearly shows that the use of multiple centrality correctors leads to a reduction in the number

TABLE VI Efficiency of multiple centrality corrections (IBM, Power 2, 58 MHz).

Correctors	AUG2DC		AUG3DC		CVXQP1	
	iters	time	iters	time	iters	time
0	11	87.2	25	4973.2	13	5221.6
1	10	80.1	23	4589.0	12	4812.6
2	**9**	75.0	23	4654.3	11	4456.4
4	9	75.6	21	4350.0	10	4105.0
6	9	75.7	19	3967.6	10	4119.4
8	9	76.6	18	3816.0	10	4097.0
10	9	78.3	**17**	3628.8	**10**	4186.1

of iterations performed by the primal-dual algorithm. For problems with difficult Cholesky factorization (AUG3DC and CVXQP1) this technique offers considerable savings in the overall solution time resulting from the decrease in the number of iterations. These results are consistent with the earlier ones reported in [14] for the primal-dual method applied in linear programming.

6.4 Comparison with LOQO

The motivation to use dynamic primal and dual regularizations presented in this paper comes from the potential advantages of finding the optimal reordering of matrix H based solely on its sparsity structure. An alternative approach implemented in Vanderbei's LOQO [27] uses a tiered elimination in which most of the pivots are forced to come primarily from either the upper-left diagonals or the lower-right diagonals in matrix (36). This tiered elimination scheme is the default option in LOQO, but it may be changed by setting `stablty` option to 0. Namely, this parameter set to 1 (default) means total separation of blocks from which pivots are to be chosen. Setting it to 0 disables the multi-tiered approach and provides an ordering of the entire matrix H_V of (36) based solely on sparsity issues.

We compared HOPDM with LOQO run with two different settings of the `stablty` parameter. This comparison was done on a PC with a 200 MHz Pentium Pro processor and 128 MB of RAM run under the Linux operating system. We tested the solvers on difficult LP problems, NUG (known to cause significant fill-in in the Cholesky factors) and a family of quadratic programs from the CUTE collection. The later were chosen for two reasons. Firstly, their sizes change, which facilitates showing the solvers' ability to deal with small, medium and larger problems. Secondly, they have a nondiagonal matrix Q and, to the best of our knowledge, they cannot easily be reformulated as separable QP problems.

In Table VII, we report problem sizes and compare the statistics on Cholesky factors obtained with both solvers. nz(Q) is the number of off-diagonal elements in the triangular part of matrix Q, nz(L) is the number of off-diagonal elements in the Cholesky matrix. LOQO's results are given for two different settings of `stablty` option. (Following the manual of LOQO, the remaining parameters of LOQO used to solve quadratic programs are: `convex, bndpush=100,`

TABLE VII Sparsity of Cholesky factors.

Problem	Dimensions				LOQO: nz(L)		HOPDM
	m	n	nz(A)	nz(Q)	stablty=1	stablty=0	nz(L)
nug05	210	225	1190	0	12968	13990	11835
nug06	372	486	2532	0	39725	39174	33511
nug07	602	931	4886	0	101590	103393	81437
nug08	912	1632	8304	0	222210	222210	179790
nug12	3192	8856	44244	0	3091223	3091223	1969957
nug15	6330	22275	110700	0	–	–	7374972
cvxqp1_s	50	100	148	286	3184	1578	1549
cvxqp1_m	500	1000	1498	2984	205980	71487	75973
cvxqp1_l	5000	10000	14998	29984	–	4056820	3725045
cvxqp2_s	25	100	74	286	1691	1246	1265
cvxqp2_m	250	1000	749	2984	88676	52917	51923
cvxqp2_l	2500	10000	7499	29984	4859668	2923584	2754141
cvxqp3_s	75	100	222	286	5337	1760	1847
cvxqp3_m	750	1000	2247	2984	399227	79957	90433
cvxqp3_l	7500	10000	22497	29984	–	4411197	4291057

`honor_bounds=0, pred_corr=1, mufactor=0.`) HOPDM is always run with the default options.

From the results collected in Table VII we conclude that the choice of stability parameter merely alters the quality of reordering for LP problems from the NUG family. Instead, when quadratic programs are solved, this parameter causes a remarkable change in the sparsity of the Cholesky matrices. The quality of reorderings generated by LOQO with `stablty` option set to 0 is comparable with that of HOPDM. The use of (default) tiered elimination results in producing remarkably denser factors.

In Table VIII, we report the solution statistics. Empty spaces corresponding to certain runs of LOQO mean that there was not enough memory to solve a given problem. From the analysis of data collected in Table VIII we can see that HOPDM solved all problems in significantly less iterations than LOQO. This translates into important CPU time savings.

One reason for the difference in the number of iterations is certainly the use of multiple centrality correctors in HOPDM. However, this cannot be the only explanation of the important differences between HOPDM and LOQO.

The weak overall regularization used in HOPDM (and concentrating the regularization only on the "dangerously small" pivots) seems to perturb the problem less than the regularization resulting from adding slack variables in the primal and dual spaces applied by LOQO, see (34)–(35).

TABLE VIII Solution Statistics.

Problem	LOQO				HOPDM	
	stablty=1		stablty=0			
	iters	time	iters	time	iters	time
nug05	13	0.64	13	0.81	7	0.62
nug06	14	3.05	14	3.47	7	1.88
nug07	17	16.23	17	17.69	9	7.99
nug08	15	50.91	15	50.12	8	22.35
nug12	24	4450.5	24	4417.7	13	1140.3
nug15	–	–	–	–	15	10276.6
cvxqp1_s	13	0.13	13	0.09	8	0.23
cvxqp1_m	32	72.88	32	13.78	10	7.33
cvxqp1_l	72	–	72	18361.1	24	5986.4
cvxqp2_s	11	0.04	11	0.04	9	0.15
cvxqp2_m	16	8.48	16	4.06	9	4.02
cvxqp2_l	25	8495.8	25	3849.4	9	1482.7
cvxqp3_s	13	0.22	13	0.09	10	0.26
cvxqp3_m	49	346.75	49	25.45	10	9.91
cvxqp3_l	–	–	100	27447.6	10	2940.2

7 CONCLUSIONS

We have described in this paper a technique for the regularization of the augmented system factorization applicable to interior point code for linear and quadratic optimization. The regularization can be interpreted in terms of the proximal point algorithm applied to the primal and dual problem, and it is particularly well suited to the primal-dual interior point method.

The regularization technique proposed here dynamically adjusts the regularization terms and can be incorporated easily into any implementation of the symmetric triangular factorization. When applied within HOPDM code, it proved to be useful in the solution of very large linear and quadratic programs.

Acknowledgements

We are grateful to Csaba Mészáros and Hans Mittelmann for providing us with their collections of large scale linear and quadratic programming problems. We are grateful to Michael Saunders, Robert Vanderbei and to the two anonymous referees for their constructive comments and suggestions that helped us to improve the final version of this paper. We also thank Crawford Buchanan for careful reading of this paper.

Anna Altman (1954–1999)

Anna Altman died on Saturday 24th July, 1999. She was suffering from a cancer, known since 1993.

She was an Assistant Professor at the Systems Research Institute. She had a broad interest in optimization varying from linear and quadratic programming to convex nonlinear programming. She published a number of articles in this area.

She was dedicated to her work, family, and friends. She was a good friend I will miss.

Jacek Gondzio

References

[1] Altman, A. (1995). QHOPDM –A higher order primal-dual method for large scale separable convex quadratic programming, *European Journal of Operational Research*, **87**, 200–202.

[2] Altman, A. (1996). Higher order primal-dual interior point method for separable convex quadratic optimization, *Control and Cybernetics*, **25**(4), 761–772.

[3] Altman, A. Amann, M. Klaassen, G. Ruszczyński, and Schöpp, W. (1996). Cost-effective sulphur emission reduction under uncertainty, *European Journal of Operational Research*, **90**, 395–412.

[4] Andersen, E. Gondzio, J. Mészáros, C. and Xu, X. (1996). Implementation of interior point methods for large scale linear programming, In T. Terlaky, editor, Kluwer Academic Publishers, *Interior Point Methods in Mathematical Programming*, pp. 189–252.

[5] Arioli, M. Demmel, J. W. and Duff, I. S. (1989). Solving sparse linear systems with sparse backward error, *SIAM Journal on Matrix Analysis and Applications*, **10**(2), 165–190.

[6] Arioli, M. Duff, I. S. and de Rijk, P. P. M. (1989). On the augmented system approach to sparse least-squares problems, *Numerische Mathematik*, **55**, 667–684.

[7] Bunch J. R. and Parlett, B. N. (1971). Direct methods for solving symemtric indefinite systems of linear equations, *SIAM Journal on Numerical Analysis*, **8**, 639–655.

[8] Carpenter, T. J. Lustig, I. J. Mulvey, J. M. and Shanno, D. F. (1993). Higher order predictor-corrector interior point methods with application to quadratic programming, *SIAM Journal on Optimization*, **3**, 696–725.

[9] Fiacco, A. V. and McCormick, G. P. (1968). *Nonlinear Programming, Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, New York.

[10] Fourer, R. and Mehrotra, S. (1993). Solving symmetric indefinite systems in an interior-point method for linear programming, *Mathematical Programming*, **62**, 15–39.

[11] Gay, D. M. (1985). Electronic mail distribution of linear programming test problems, *Mathematical Programming Society COAL Newsletter*, **13**, 10–12.

[12] Gondzio, J. (1993). Implementing Cholesky factorization for interior point methods of linear programming, *Optimization*, **27**, 121–140.

[13] Gondzio, J. (1995). HOPDM (version 2.12) — a fast LP solver based on a primal-dual interior point method, *European Journal of Operational Research*, **85**, 221–225.

[14] Gondzio, J. (1996). Multiple centrality corrections in a primal-dual method for linear programming, *Computational Optimization and Applications*, **6**, 137–156.

[15] Kojima, M. Mizuno, S. and Yoshise, A. (1989). A primal-dual interior point algorithm for linear proramming. In N. Megiddo, editor, *Progress in Mathematical Programming, Interior-Point Algorithms and Related Methods*, pp 29–47. Springer Verlag, Berlin, Germany.

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

[17] Lustig, I. J. Marsten, R. E. and Shanno, D. F. (1994). Interior point methods for linear programming, computational state of the art, *ORSA Journal on Computing*, **6**, 1–14.

[18] Maros, I. and Mészáros, C. (1997). A repository of convex quadratic programming problems, Technical Report DOC 97/6, Department of Computing, Imperial College, London, U.K., July.

[19] Megiddo. N. (1989). Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming, Interior-Point Algorithms and Related Methods*, Springer Verlag, Berlin, Germany.

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

[21] Mittelmann, H. (1998). Decision tree for optimization software, January.

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

[23] Rockafellar. R. T. (1976). Monotone operators and the proximal point algorithm in convex programming, *SIAM Journal on Control and Optimization*, **14**, 887–898.

[24] Saunders. M. (1996). Cholesky-based methods for sparse least squares, the benefits of regularization, In L. Adams and L. Nazareth, editors, *Linear and Nonlinear Conjugate Gradient-Related Methods*, pp 92–100. SIAM, Philadelphia, USA.

[25] Saunders, M. and Tomlin. J. A. (1996). Solving regularized linear programs using barrier methods and KKT systems, Technical Report SOL 96-4, Systems Optimization Laboratory, Dept. of Operatoions Research, Stanford University, Stanford, CA 94305, USA, December.

[26] Setiono. R. (1992). Interior proximal point algorithm for linear programs. *Journal of Optimization Theory and Application*, **74**, 425–444.

[27] Vanderbei. R. J. (1994). LOQO: An interior point code for quadratic programming, Technical Report SOR-94-15, Statistics and Operations Research, Princeton University.

[28] Vanderbei. R. J. (1995). Symmetric quasidefinite matrices, *SIAM Journal on Optimization*, **5**, 100–113.

[29] Vanderbei R. J. and Carpenter. T. J. (1993). Symmetric indefinite systems for interior point methods, *Mathematical Programming*, **58**, 1–32.

[30] Wright. S. (1997). *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia.