Solving Sparse Convex Quadratic Programming Problems with the C Numerical Library

Mathematical and Numerical Aspects of the Sparse Convex QP Solver sparse_quadratic_prog

A White Paper by Rogue Wave Software.
Solving Sparse Convex Quadratic Programming Problems with the C Numerical Library

Mathematical and Numerical Aspects of the Sparse Convex QP Solver sparse_quadratic_prog

by Rogue Wave Software

© 2012 by Rogue Wave Software. All Rights Reserved

Printed in the United States of America

Publishing History:
November 2012 – Update

November 2011 – First Publication

Trademark Information

The Rogue Wave Software name, logo, IMSL and PV-WAVE are registered trademarks of Rogue Wave Software, Inc. or its subsidiaries in the US and other countries. JMSL, JWAVE, TS-WAVE, and PyIMSL are trademarks of Rogue Wave Software, Inc. or its subsidiaries. All other company, product or brand names are the property of their respective owners.

IMPORTANT NOTICE: The information contained in this document is subject to change without notice. Rogue Wave Software, Inc. makes no warranty of any kind with regards to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Rogue Wave Software, Inc. shall not be liable for errors contained herein or for incidental, consequential, or other indirect damages in connection with the furnishing, performance, or use of this material.
# TABLE OF CONTENTS

Introduction .................................................................................................................................... 4
Problem Formulation ...................................................................................................................... 5
The Algorithm .................................................................................................................................. 5
   Infeasible Primal-Dual Interior-Point Methods ................................................................ 5
   The Mehrotra Predictor-Corrector ....................................................................................... 7
   Multiple Centrality Correctors .......................................................................................... 8
   Weighted Correctors .............................................................................................................. 10
Linear Algebra ................................................................................................................................ 12
Presolving Techniques .................................................................................................................. 15
   Elimination of Row Singletons .......................................................................................... 16
   Bounds on Primal Constraints and Variables .................................................................. 17
   Bounds on Dual Variables, Dominance and Weak Dominance ...................................... 18
      Column Singletons ....................................................................................................... 18
      Tightening Bounds on Dual Variables and Fixing Primal Variables ......................... 19
      Elimination of Column Singletons .............................................................................. 20
   Removal of Doubleton Equality Rows .......................................................................... 22
   Column Aggregation .......................................................................................................... 22
Numerical Results ......................................................................................................................... 25
Conclusion ..................................................................................................................................... 29
About the IMSL Libraries ........................................................................................................... 29
About Rogue Wave Software ...................................................................................................... 29
References .................................................................................................................................... 30
Introduction

This survey article describes in some detail the mathematical and numerical aspects of function \texttt{ims1\_d\_sparse\_quadratic\_prog} (\texttt{sparse\_quadratic\_prog} for short), the sparse convex quadratic programming solver implemented in IMSL® C Numerical Library 8.0. The solver uses an infeasible primal-dual interior-point method to find an optimal solution. All primal-dual interior-point methods are essentially variants of Newton’s method that in every iteration step solve a perturbed system of the first-order optimality conditions in order to drive the new iterate closer to the optimum. Today, such methods are widely seen as very efficient means to solve large-scale linear and convex quadratic programming problems fast and accurately.

The article is organized as follows:

- Problem Formulation
- The Algorithm
- Linear Algebra
- Presolving Techniques
- Numerical Results
- Conclusion

In the “Problem Formulation” section, the quadratic programming problem that is solved by \texttt{sparse\_quadratic\_prog}, is formulated. In “The Algorithm” section, the primal-dual interior-point method used in \texttt{sparse\_quadratic\_prog} is described in some detail.

One of the most important aspects of interior-point methods is the efficient solution of a large sparse linear system in every step of the algorithm. The “Linear Algebra” section describes the techniques used in this context. Presolving is an efficient tool to simplify the quadratic programming problem before the optimization starts. A detailed overview of the presolving methods used by \texttt{sparse\_quadratic\_prog} is given in the “Presolving Techniques” section. The “Numerical Results” section shows some numerical results obtained by applying \texttt{sparse\_quadratic\_prog} to some of the test problems in the convex quadratic programming repository of [10]. Finally, the “Conclusion” section draws some conclusions from the presented material and results.

The interior-point solver in \texttt{sparse\_quadratic\_prog} is based on Jacek Gondzio’s code HOPDM, a state-of-the-art infeasible primal-dual interior-point code for solving linear and quadratic optimization problems. This code was described in a series of papers ([1],[3],[5],[7],[8],[9]). The description given in “The Algorithm”, “Linear Algebra”, and “Presolving Techniques” relies heavily on these results.
Problem Formulation

Consider the following convex quadratic optimization problem with linear constraints

\[
\begin{align*}
\min & \quad f(x) = c^T x + \frac{1}{2} x^T Q x \\
\text{s.t.} & \quad b \leq Ax \leq b, \\
& \quad l \leq x \leq u
\end{align*}
\]

where \( c, x \in \mathbb{R}^n, l \in \mathbb{R}^n \cup \{-\infty\}^n, u \in \mathbb{R}^n \cup \{+\infty\}^n, b \in \mathbb{R}^m \cup \{-\infty\}^m \) and \( \bar{b} \in \mathbb{R}^m \cup \{+\infty\}^m \) are vectors, \( A \in \mathbb{R}^{m \times n} \) is the constraint matrix and \( Q \in \mathbb{R}^{n \times n}, Q \neq 0 \), is symmetric positive semidefinite. Note that the definition of \( l, u \) and \( \bar{b}, \bar{b} \) also allows for free variables and free constraints, respectively.

Function \texttt{sparse_quadratic_prog} uses this formulation of the quadratic programming (QP) problem for presolving. Presolving is usually the first step in solving a QP problem. We will consider it in more detail in “Presolving Techniques”.

After the presolving phase \texttt{sparse_quadratic_prog} transforms the reduced problem into an equivalent form which is computationally more easily tractable. After redefining the notation, the QP problem now appears as follows:

\[
\begin{align*}
\min & \quad c^T x + \frac{1}{2} x^T Q x \\
& \quad Ax = b, \\
& \quad 0 \leq x_i \leq u_i, i \in U \\
& \quad 0 \leq x_i, i \in L
\end{align*}
\]

Here, \( b \in \mathbb{R}^n, u_i \in \mathbb{R} \) with \( u_i \geq 0 \) for \( i \in U \), where the sets \( U \) and \( L \) describe a partition of the index set \( \{1, \ldots, n\}, U \cup L = \{1, \ldots, n\}, U \cap L = \emptyset \).

The Algorithm

\textit{Infeasible Primal-Dual Interior-Point Methods}

In order to simplify the following mathematical presentation, henceforth let’s assume that all variables in the QP problem are upper bounded. By introducing a vector \( s \) of slack variables, the problem then has its final form

\[
\begin{align*}
\min & \quad f(x) = c^T x + \frac{1}{2} x^T Q x \\
\text{s.t.} & \quad Ax = b, \\
& \quad x + s = u, \\
& \quad x, s \geq 0
\end{align*}
\]
The dual of problem \((P)\) can be written as
\[
\begin{align*}
\max & \quad \varphi(x, y, w, z) = -\frac{1}{2} x^T Q x + b^T y - u^T w \\
(D) & \quad A^T y + z - w - Q x = c, \\
& \quad w, z \geq 0
\end{align*}
\]

Using the Lagrangian for problem \((P)\),
\[
L(x, s, y, \lambda, z, w) = c^T x + \frac{1}{2} x^T Q x - y^T (A x - b) - \lambda^T (x + s - u) - z^T x - w^T s,
\]

it is easy to verify that the Karush-Kuhn-Tucker (KKT) conditions for \((P)\) and \((D)\) are

\[
\begin{align*}
Ax & = b, \quad (1) \\
x + s & = u, \quad (2) \\
A^T y + z - w - Q x & = c, \quad (3) \\
XZe & = 0, \quad (4) \\
SWe & = 0, \quad (5) \\
x, s, z, w & \geq 0. \quad (6)
\end{align*}
\]

Here, the matrices \(X, S, Z, W \in \mathbb{R}^{n \times n}\) are diagonal matrices with the entries of vectors \(x, s, z\) and \(w\), respectively, on their diagonals, and \(e = (1, \ldots, 1)^T \in \mathbb{R}^n\) is a vector of ones.

Whereas equations (1), (2), (3) and (6) guarantee primal and dual feasibility, equations (4) and (5) are the complementarity conditions. In path-following interior-point methods the complementarity conditions are replaced by the perturbed systems

\[
XZe = \mu e \quad \text{and} \quad SWe = \mu e,
\]

where \(\mu > 0\) is a barrier parameter and \((x, s, z, w)\) is enforced to be \(> 0\).

In every step, interior-point methods try to solve the nonlinear system

\[
F(x, s, y, z, w) = 
\begin{bmatrix}
Ax - b \\
x + s - u \\
A^T y + z - w - Q x - c \\
XZe - \mu e \\
SWe - \mu e
\end{bmatrix} = 0
\]

for \((x, s, z, w) > 0\) and a given specific \(\mu > 0\). The generated sequence \(\{\mu_k\}\) of barrier parameters tends to zero, \(\mu_k \downarrow 0\) for \(k \to \infty\). Infeasible interior-point-methods in particular generate iterates
that satisfy \((x_k, s_k, y_k, z_k, w_k) > 0\) for all \(k\), but are in general infeasible, i.e. the primal and dual linear constraints (1)-(3) are only satisfied for \(k \rightarrow \infty\).

For a given \((x, s, y, z, w)\), where \((x, s, z, w) > 0\), and barrier parameter \(\mu > 0\), the nonlinear system (7) is linearized using Newton’s method:

\[
\nabla F(x, s, y, z, w) (\Delta x, \Delta s, \Delta y, \Delta z, \Delta w) = -F(x, s, y, z, w).
\]

This leads to the so-called step equations

\[
\begin{bmatrix}
A & 0 & 0 & 0 & 0 \\
I & I & 0 & 0 & 0 \\
-Q & 0 & A^T & I & -I \\
Z & 0 & 0 & X & 0 \\
0 & W & 0 & 0 & S
\end{bmatrix}
\begin{bmatrix}
\Delta x \\
\Delta s \\
\Delta y \\
\Delta z \\
\Delta w
\end{bmatrix} = r,
\]

where

\[
r = (b - Ax, u - x - s, c - A^T y - z + w + Qx, -XZe + \mu e, -SWe + \mu e).
\]

Once the Newton directions \((\Delta x, \Delta s)\) and \((\Delta y, \Delta z, \Delta w)\) in the primal and dual space, respectively, are computed, the maximum stepsize \(\alpha\) that preserves nonnegativity of the variables in the primal and dual spaces is determined:

\[
\alpha = \max \{ \lambda \in [0,1] : (x, s, z, w) + \lambda (\Delta x, \Delta s, \Delta z, \Delta w) \geq 0 \}.
\]

Using a step-reduction factor \(\alpha_0 = 0.99995\), the next iterate is then computed:

\[
(x_{k+1}, s_{k+1}, y_{k+1}, z_{k+1}, w_{k+1}) = (x_k, s_k, y_k, z_k, w_k) + \alpha_0 \alpha (\Delta x, \Delta s, \Delta y, \Delta z, \Delta w).
\]

The Mehrotra Predictor-Corrector

Mehrotra’s predictor-corrector technique decomposes the computation of the Newton direction \(\Delta := (\Delta x, \Delta s, \Delta y, \Delta z, \Delta w)\) in (8) into two parts,

\[
\Delta = \Delta_a + \Delta_c,
\]

where \(\Delta_a\) is the affine-scaling predictor direction and \(\Delta_c\) Mehrotra’s second order centering direction. The term \(\Delta_a\) is obtained by solving the linear system (8) for right-hand side
\[ r_i = (b - Ax, u - x - s, c - A^Ty - z + w + Qx, -XZe, -SWe). \]  

(11)

The affine-scaling step is responsible for optimization because it tries to find a solution to the KKT conditions (1)-(6). If \((x, s, y, z, w) + \Delta_x\) is not optimal then a centering component \(\Delta_c\) has to be computed.

To this purpose, the maximum stepsize \(\alpha\) that preserves nonnegativity of both \((x, s)\) and \((z, w)\) is determined and used to compute the predicted complementarity gap

\[ g_a = (x + \alpha \Delta_x x)^T (z + \alpha \Delta_z z) + (s + \alpha \Delta_s s)^T (w + \alpha \Delta_w w). \]

This value is then used to determine the barrier parameter

\[ \mu = \left( \frac{g_a}{g} \right)^2 \frac{2n}{g_a} \]  

(12)

where \(g = x^T z + s^T w\) denotes the current complementarity gap.

System (8) is now solved for the right-hand side

\[ r_z = (0, 0, 0, \mu e - \Delta_x X \Delta_x Ze, \mu e - \Delta_s S \Delta_s We). \]  

(13)

to obtain the corrector direction \(\Delta_c\). The purpose of direction \(\Delta_c\) is to keep the (hypothetical) iterate \((x, s, y, z, w) + \Delta_x\) away from the boundary given by (6) and to drive the new iterate back to the central path.

Finally, by adding both directions, (10), we get the Mehrotra predictor-corrector direction.

Note that the computation of the predictor-corrector direction requires the solution of a probably large sparse linear system (8) for two different right-hand sides. The decomposition of the left-hand side matrix in (8) can therefore be used twice to determine \(\Delta_x\) and \(\Delta_c\).

**Multiple Centrality Correctors**

Assume \((x, s)\) and \((y, z, w)\) are the current primal and dual iterates of the interior-point method described in subsection "Infeasible Primal-Dual Interior-Point Methods". Assume also that a predictor direction \(\Delta_p\) and the associated stepsize \(\alpha\) that preserves nonnegativity of the primal and dual variables have been computed. The goal now is to find a centrality corrector \(\Delta_m\) that allows for a larger stepsize in the composite direction.
\[ \Delta = \Delta_p + \Delta_m. \] (14)

Specifically (see [5]), the stepsize \( \alpha \) should be enlarged to

\[ \hat{\alpha} = \min(\alpha + \delta, 1) \]

for some aspiration level \( \delta \in (0,1) \).

The trial points

\[ (\tilde{x}, \tilde{s}) = (x, s) + \hat{\alpha}(\Delta_p x, \Delta_p s), \quad (\tilde{z}, \tilde{w}) = (z, w) + \hat{\alpha}(\Delta_p z, \Delta_p w) \] (15)

and the corresponding complementarity products

\[ (\tilde{v}_{xz}, \tilde{v}_{zw}) = (\tilde{XZ}e, \tilde{ZW}e) \in \mathbb{R}^{2n} \] (16)

are computed. We now try to drive the trial points (15) back into the vicinity of the central path by defining the targets

\[ t^x_j = \begin{cases} 
\gamma \mu - \tilde{x}_j \tilde{z}_j & \text{if } \tilde{x}_j \tilde{z}_j \leq \gamma \mu \\
\gamma^{-1} \mu - \tilde{x}_j \tilde{z}_j & \text{if } \tilde{x}_j \tilde{z}_j \geq \gamma^{-1} \mu \\
0 & \text{otherwise}
\end{cases} \]

\[ t^w_j = \begin{cases} 
\gamma \mu - \tilde{s}_j \tilde{w}_j & \text{if } \tilde{s}_j \tilde{w}_j \leq \gamma \mu \\
\gamma^{-1} \mu - \tilde{s}_j \tilde{w}_j & \text{if } \tilde{s}_j \tilde{w}_j \geq \gamma^{-1} \mu \\
0 & \text{otherwise}
\end{cases} \] (17)

Here, \( \gamma \in (0,1) \) is a parameter defining the neighborhood of the central path.

The corrector \( \Delta_m \) is then computed as the solution to step equation (8) for right-hand side

\[ r = (0,0,0,t^x,t^w) \] (18)

Adding \( \Delta_m \) to \( \Delta_p \) then finally gives the corrected search direction \( \Delta \) in (14).

For the new search direction \( \Delta \), the maximum step length \( \hat{\alpha} \) in the primal and dual space is determined and equation (9) is used to compute the next iterate.

The correction process can be made iterative by taking \( \Delta \) as the new predictor direction, \( \Delta_p := \Delta \). Next, steps (15), (16) and (17) are repeated to get a new target vector \( r \) which is used as right-hand side in (8) to get the new correction \( \Delta_m \), which in turn is added to \( \Delta_p \), giving a new search direction (14).
When computing centrality correctors iteratively it is necessary to define a criterion when the process of adding correction terms \( \Delta_m \) should be stopped. Assume that the maximum stepsize for a given predictor direction is \( \alpha \) and the maximum stepsize for the corrected direction \( \Delta \) in (14) is \( \hat{\alpha} \). Then, \( \Delta \) is accepted as new predictor direction \( \Delta_p \), if

\[
\hat{\alpha} \geq \alpha + \theta \delta,
\]

where \( \theta \in (0,1) \) is a given tolerance value. Otherwise, the correction process is stopped.

**Weighted Correctors**

In order to limit or adjust the influence of the corrector, the corrector direction in the Mehrotra predictor-corrector (10) can be weighted:

\[
\Delta^\omega = \Delta_a + \omega \Delta_c.
\]  

Here, the weight \( \omega \in [\omega_{\text{min}},1] \) with a specified \( \omega_{\text{min}} \in (0,1) \) is chosen independently at each iteration so that the steplength \( \alpha^\omega \) in direction \( \Delta^\omega \) is maximized:

\[
\alpha^\omega = \max \{ \alpha \in [0,1] : (x,s,y,z,w) + \alpha (\Delta_a + \omega \Delta_c) \geq 0, \omega \in [\omega_{\text{min}},1] \}.
\]

For the optimal weight \( \hat{\omega} = 1 \), \( \Delta^\omega \) coincides with the Mehrotra predictor-corrector again.

In a similar way, the weighted corrector strategy can be applied to the multiple centrality correctors as well.

We have now described all ingredients that are used in the primal-dual method implemented in function `sparse_quadratic_prog` (see [5]):

**Given** a starting point \((x^0, s^0, y^0, z^0, w^0)\) with \((x^0, s^0, z^0, w^0) > 0\), the maximum number \(M\) of centrality corrections allowed at each iteration, the aspiration level \( \delta \in (0,1) \), \( \gamma \in (0,1) \), \( \theta \in (0,1) \), \( \omega_{\text{min}} \in (0,1) \) and \( \alpha = 0.99995 \)

**for** \( k = 0,1,2,\ldots \)

- If some convergence criteria are satisfied, then **stop**;

**Mehrotra predictor-corrector step:**

- **Set** \((x, s, y, z, w) = (x^k, s^k, y^k, z^k, w^k)\) and solve system (8) with right-hand side \( r_1 \) in (11) for an affine-scaling direction \( \Delta_a \).

- **Set** \( \mu \) as defined in (12) and solve system (8) with right-hand side \( r_2 \) in (13) for Mehrotra’s corrector direction \( \Delta_c \).
• Do a line-search to find the optimal $\hat{\omega}$ that maximizes the stepsize $\alpha$ in $\Delta^{\alpha} = \Delta_a + \omega \Delta_c$. Denote the maximum stepsize by $\alpha_k$.

Multiple centrality correctors:

• Set $\Delta_p = \Delta_a + \hat{\omega} \Delta_c$ as the predictor direction; set $l = 0$.

\[
\text{do} \\
\quad \text{Set } l := l + 1. \\
\quad \text{if } l > MC_{\max} \text{ then break; } \\
\quad \quad \text{Solve system (8) with right-hand side (18), } t^{\omega}, t^{\omega w} \text{ defined as in (17), for a centrality corrector direction } \Delta_m. \\
\quad \quad \text{Do a line-search to find the optimal } \hat{\omega} \text{ that maximizes the stepsize } \alpha \text{ in direction } \Delta^{\alpha} = \Delta_p + \omega \Delta_m. \text{ Denote the maximum stepsize by } \hat{\alpha}. \\
\quad \quad \text{if } \hat{\alpha} \geq \alpha_k + \theta \delta \text{ (see (19)) then } \\
\quad \quad \quad \text{Set } \Delta_p := \Delta_p + \hat{\omega} \Delta_m \text{ and } \alpha_k := \hat{\alpha}. \\
\quad \quad \text{else} \\
\quad \quad \quad \text{break;} \\
\quad \text{endif} \\
\text{end do}
\]

Update:

• Update the iterate: $$(x^{k+1}, s^{k+1}, y^{k+1}, z^{k+1}, w^{k+1}) = (x^k, s^k, y^k, z^k, w^k) + \alpha \Delta_p$$

A heuristic for finding $MC_{\max}$, the maximum number of centrality correctors, is explained in more detail in the “Linear Algebra” section below.

Function `sparse_quadratic_prog` stops with optimal termination status if the current iterate $(x^k, s^k, y^k, z^k, w^k, \mu^k)$ satisfies the following three convergence criteria:

\[
\frac{\mu}{1 + 0.5(|c^T x| + |b^T y - u^T w - 0.5x^T Qx|)} \leq \text{opt_tol} \\
\|b - Ax, x + s - u\|_2 \leq \text{prinf_tol},
\]

www.roguewave.com
where \(\text{prinf}_\text{tol}, \text{dlinf}_\text{tol}\) and \(\text{opt}_\text{tol}\) denote primal infeasibility, dual infeasibility and optimality tolerances, respectively. The default value in \text{sparse}_\text{quadratic}_\text{prog} is \(1.0\times10^{-10}\) for \(\text{opt}_\text{tol}\) and \(1.0\times10^{-8}\) for the two other tolerances.

We want to stress that the primal-dual algorithm was derived under the theoretically correct assumption that primal-dual interior-point methods applied to convex quadratic programming problems require the stepsizes for the Newton directions in the primal and dual space to be the same. This is in contrast to linear programs where the primal and dual stepsizes can be chosen independently. However, intensive practical testing has shown, that even in the convex QP case different stepsizes can be chosen. In general, this leads to faster convergence at the expense of some accuracy of the solution. However, especially on large scale QPs the advantage of fewer iterations seems to outweigh the often only small loss in accuracy.

Therefore, actually, in function \text{sparse}_\text{quadratic}_\text{prog} a simple modification of the primal-dual algorithm above is implemented that always chooses primal and dual stepsizes independently from each other.

**Linear Algebra**

By elimination of \(\Delta s, \Delta z\) and \(\Delta w\), the linear system (8) can be reduced to a more compact form, the so-called \textit{augmented system} (AS), whose coefficient matrix is given by

\[
H \equiv \begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix}
\]  

or further by elimination of \(\Delta x\) to the \textit{normal equations} (NE) system with coefficient matrix

\[
M \equiv A(Q + \Theta^{-1})^{-1} A^T
\]

where \(\Theta = (X^{-1}Z + S^{-1}W)^{-1}\) is a positive definite diagonal matrix.

Early implementations of interior-point methods mostly used the NE approach to solve system (8). The advantage of this approach is that matrix \(M\) is positive definite so that well-known direct and iterative methods like the Cholesky factorization and the preconditioned conjugate gradient (PCG) method, respectively, can be applied to such systems. The disadvantage however is that, even if matrix \(A\) is sparse, matrix \(M\) might be quite dense. This is, for example, the case if matrix \(A\) contains dense columns.
In comparison to the NE approach the AS approach has some advantages. For example, the augmented system with coefficient matrix $H$ is much more stable than the NE system with coefficient matrix $M$ and the presence of dense columns in $A$ does not increase the fill-in significantly. Since matrix $H$ is symmetric indefinite, the solution of the AS requires a Bunch-Parlett factorization of (21). However, the Bunch-Parlett factorization is usually computationally more expensive than the Cholesky factorization of (22), because here the sparsity preserving reordering cannot be separated from the numerical factorization. This problem can be solved by transforming $H$ into a symmetric quasidefinite matrix, i.e. a matrix of the form

$$
\begin{bmatrix}
-E & A^T \\
A & F
\end{bmatrix}
$$

where $E$ and $F$ are positive definite matrices. Symmetric quasidefinite matrices are nonsingular and strongly factorizable (see [11]): If $K$ is symmetric quasidefinite, then for every permutation matrix $P$ a factorization

$$
PKP^T = LDL^T
$$

exists with $D$ a diagonal matrix and $L$ a unit lower triangular matrix. Hence, matrix $K$ allows for a generalized Cholesky factorization in which the reordering can be done strictly before the factorization.

Within function `sparse_quadratic_prog`, the transformation of matrix $H$ into a quasidefinite matrix is realized by adding regularization terms to the diagonals, i.e. $H$ is replaced by

$$
H' \equiv \begin{bmatrix}
-Q - \Theta^{-1} & A^T \\
A & 0
\end{bmatrix} + \begin{bmatrix}
-R_p & 0 \\
0 & R_d
\end{bmatrix}
$$

(23)

where the regularizations $R_p$ and $R_d$ are diagonal matrices with (small) positive diagonal elements. These elements are chosen dynamically during the factorization phase of the (possibly) reordered matrix $H'$.

For more information on this regularization technique, the interested reader is referred to [1]. Though the authors give no convergence proof for their method it seems to work well in practice, see e.g. the numerical results in [1].

Function `sparse_quadraticProg` uses the AS approach with regularized matrix $H$ to solve system (8) either by a direct method (generalized Cholesky) or by an iterative method (Projected Conjugate Gradient (PCG)). If the number of constraints or the number of nonzero entries in matrix $H$ is small then the direct method is preferred. Otherwise, the iterative approach is used.
We return to the problem of the determination of the maximum number of multiple correctors $MC_{\text{max}}$ described above in the “The Algorithm” section. First, if the PCG - method is used to solve the AS, then $MC_{\text{max}}$ is set to 2. If the direct approach is used then the number $MC_{\text{max}}$ depends on the computational effort that is needed to factorize matrix (23), $H' = LDL^T$, and the effort needed to solve the AS

\[
LDL^T \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}
\]

(24)

for an arbitrary right-hand side $(r_1, r_2)$. According to [9], the factorization effort $E_f$ and the effort $E_s$ to solve (24) can be measured by

\[
E_f = \sum_{i=1}^{m} l_i^2 \quad \text{and} \quad E_s = 2*\sum_{i=1}^{m} l_i + 12*n
\]

where $l_i$ is the number of off-diagonal nonzero entries in $L$ and $m, n$ are the sizes of the LP problem. (Note that since the sparsity structure of $H'$ is the same in all iterations of the interior-point algorithm, a sparsity preserving reordering and the symbolic factorization of $H'$ has to be done only once, before the interior point algorithm starts. Hence, the number of entries in the columns of $L$ does not change from iteration to iteration.)

Clearly, the larger the ratio $r_{f/s} = E_f / E_s$, the more centrality correctors should be allowed by solving the same system (24) for different right-hand sides. The following table shows from practical experience which values of $MC_{\text{max}}$ should be chosen for all possible values of $r_{f/s}$:

<table>
<thead>
<tr>
<th>Range</th>
<th>$MC_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_{f/s} &lt; 8$</td>
<td>3</td>
</tr>
<tr>
<td>$8 \leq r_{f/s} &lt; 30$</td>
<td>4</td>
</tr>
<tr>
<td>$30 \leq r_{f/s} &lt; 50$</td>
<td>5</td>
</tr>
<tr>
<td>$50<em>p \leq r_{f/s} &lt; 50</em>(p+1), \ p = 1, ..., 5$</td>
<td>$p+5$</td>
</tr>
<tr>
<td>$100<em>p \leq r_{f/s} &lt; 100</em>(p+1), \ p = 3, 4$</td>
<td>$p+8$</td>
</tr>
<tr>
<td>$r_{f/s} \geq 500$</td>
<td>13</td>
</tr>
</tbody>
</table>

Table 1: Correlation between $r_{f/s}$ and $MC_{\text{max}}$

Function `sparse_quadratic_prog` uses the correlation in Table 1 to assign appropriate values for $MC_{\text{max}}$. 

www.roguewave.com
Presolving Techniques

This section discusses techniques that analyze a given LP or QP problem prior to the solution phase. The aim is to simplify the problem by reducing the problem size and to detect infeasibility or unboundedness of the problem. In `sparse_quadratic_prog`, several presolving techniques are implemented.

For the present section and following subsections, we first have to extend our notation: For a given matrix $A$, $a_{ij}$ denotes the entry of $A$ in row $i$ and column $j$, $a_i$ denotes the $i$-th row of $A$ and $a_j$ the $j$-th column of $A$. For a vector $v$, $(v)_i$ denotes its $i$-th component. If $Q \in \mathbb{R}^{n \times n}$, then $Q \succeq 0$ means that $Q$ is symmetric positive semidefinite.

We start again with the original QP formulation (QP) in “Problem Formulation”. The first order optimality conditions for (QP) are given by

\begin{align*}
    c + Qx - A^T y - z &= 0, \\
    b &\leq Ax \leq \bar{b} \\
    l &\leq x \leq u \\
    \begin{cases}
        \geq 0 & (Ax)_i \geq (Ax)_i = \frac{b_i}{\bar{b}_i} \neq (Ax)_i = \frac{b_i}{\bar{b}_i} = \frac{\infty}{\infty} \\
        = 0 & (Ax)_i < (Ax)_i = \frac{b_i}{\bar{b}_i} \neq (Ax)_i = \frac{b_i}{\bar{b}_i} = \frac{\infty}{\infty} \\
        \leq 0 & (Ax)_i = (Ax)_i = \frac{b_i}{\bar{b}_i} \neq (Ax)_i = \frac{b_i}{\bar{b}_i} = \frac{\infty}{\infty}
    \end{cases}
\end{align*}

(25) (26) (27) (28)

Conversely, for optimal solutions $x$, $y$ and $z$ the complementarity conditions lead to the following implications:

\begin{align*}
    u_i = +\infty &\land l_i = +\infty &\land z_i > 0 \implies x_i = l_i \\
    l_i = -\infty &\land u_i = +\infty &\land z_i < 0 \implies x_i = u_i
\end{align*}

(30) (31)
Based on these conditions, function `sparse_quadratic_prog` employs several presolve techniques that will be described in the following subsections. In the following we will frequently make use of the fact that for a positive semidefinite matrix that contains a zero on its diagonal the corresponding row and column is also equal to zero:

\[
Q \geq 0 \land q_{ii} = 0 \Rightarrow q_{ix} = 0 \land q_{xj} = 0.
\]  

(34)

Consequently, if \( q_{ix} = 0 \), variable \( x_i \) appears only linearly in the objective function \( f(x) \) of (QP).

**Elimination of Row Singletons**

A row singleton is a row \( i \) in the constraint matrix \( A \) that contains only one non-zero entry:

\[
\exists j: a_{ij} \neq 0, a_{ik} = 0 \; \forall k \neq j
\]

From (26) it now follows for the case \( a_{ij} > 0 \) (case \( a_{ij} < 0 \) is similar) that

\[
\bar{b}_j = \frac{b}{a_{ij}} \leq x_i \leq \frac{\bar{b}_j}{a_{ij}} = \bar{b}_j
\]

If constraint \( i \) is an equality constraint, i.e. \( \bar{b}_j = \bar{b}_i \), then variable \( x_j \) can be fixed at value \( \bar{b}_j \) if \( \bar{b}_j \) satisfies the box constraint in (27), \( l_j \leq \bar{b}_j \leq u_j \). Otherwise problem (QP) is infeasible.

If constraint \( i \) is of inequality type then the new bounds \( \bar{b}_j' \) and \( \bar{b}_i' \) can be redundant (\( \bar{b}_j' \leq l_j \) and \( \bar{b}_i' \geq u_j \)), tightening (\( \bar{b}_j < \bar{b}_j' \leq u_j \) or \( l_j \leq \bar{b}_i' < \bar{u}_j \)) or indicate that problem (QP) is infeasible (\( \bar{b}_i' > u_j \) or \( \bar{b}_i' < l_j \)).

In any feasible case, the singleton row can be eliminated from (QP). If variable \( x_j \) can be fixed, then after updating the objective function \( f(x) \), it can also be removed from the problem.

Note that the elimination of singleton rows may generate further singleton rows that can subsequently be eliminated.

If function `sparse_quadratic_prog` is called with a presolve level of at least 1, then all singleton rows are eliminated from the problem.
Bounds on Primal Constraints and Variables

For a given constraint $i$ in (26),

$$b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \overline{b}_i,$$  \hspace{1cm} (35)

set $P_i = \{j : a_{ij} > 0\}$ and $N_i = \{j : a_{ij} < 0\}$. Further assume that the lower and upper bounds $l_k$ and $u_k$ are finite for all $k \in P_i \cup N_i$. Then exploiting the variable constraints (27), the following implicit lower and upper bounds $b'_i$ and $\overline{b}'_i$, respectively, can be derived for the $i$-th constraint:

$$b'_i = \sum_{j \in P_i} a_{ij} l_j + \sum_{j \in N_i} a_{ij} u_j,$$

$$\overline{b}'_i = \sum_{j \in P_i} a_{ij} u_j + \sum_{j \in N_i} a_{ij} l_j.$$

For any $x$ satisfying the variable constraints (27) it follows

$$b'_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \overline{b}'_i.$$  \hspace{1cm} (36)

Comparing these implied bounds with the original bounds $b_i$ and $\overline{b}_i$, several cases can occur:

- $\overline{b}'_i < b_i$ or $\overline{b}'_i > \overline{b}_i$: Problem (QP) is infeasible.
- $\overline{b}'_i = b_i$ or $\overline{b}'_i = \overline{b}_i$:
  - The constraint is forcing, i.e. it can only be satisfied by fixing the variables on their respective bounds (Case $\overline{b}'_i = b_i$: $x_j = u_j \ \forall \ j \in P_i$, $x_j = l_j \ \forall \ j \in N_i$; Case $\overline{b}'_i = \overline{b}_i$: $x_j = l_j \ \forall \ j \in P_i$, $x_j = u_j \ \forall \ j \in N_i$). The constraint and the variables can be removed from the problem.
- $b'_i \leq b_i$ and $\overline{b}'_i \leq \overline{b}_i$: The constraint is redundant, i.e. it can be removed.
- ($b'_i < b_i$ and $b'_i < \overline{b}_i$) or ($b'_i < \overline{b}_i$ and $b'_i < \overline{b}_i$):
  - The constraint cannot be removed, but one of the bounds can be tightened if $\overline{b}'_i < b_i$ or $\overline{b}'_i < \overline{b}_i$. This case is non-constructive.

The non-constructive case can be used to tighten the bounds on some of the variables. Assume for example that $b'_i < b_i$. Then, using (35) and (36), we get the following bounds:

$$\forall k \in P_i : \ \overline{b}_i \geq \sum_{j=1}^{n} a_{ij} x_j \geq a_{ik} (x_k - l_k) + b'_i \ \Rightarrow \ x_k \leq l_k + \frac{\overline{b}_i - b'_i}{a_{ik}} = u'_k,$$

$$\forall k \in N_i : \ \overline{b}_i \geq \sum_{j=1}^{n} a_{ij} x_j \geq a_{ik} (x_k - u_k) + b'_i \ \Rightarrow \ x_k \geq u_k + \frac{\overline{b}_i - b'_i}{a_{ik}} = l'_k.$$
The implied bounds \( l'_k \) and \( u'_k \) can be tightening, redundant or contradictory to the existing bounds \( l_k \) and \( u_k \) (in the latter case problem (QP) is infeasible).

The analysis above assumed finiteness of all lower and upper bounds \( l_k \) and \( u_k \) for all \( k \in P_i \cup N_{i^*} \). It can be shown (see [7]) that if exactly one \( l_k \) or \( u_k \), \( k \in P_i \cup N_{i^*} \), is infinite then an implied bound can be derived for the associated variable \( x_k \).

All the techniques of this section are applied to problem (QP) if \texttt{sparse_quadratic_prog} is called with a presolve level of at least 2.

**Bounds on Dual Variables, Dominance and Weak Dominance**

Throughout this section it is always assumed that if a column \( k \) of \( A \) is considered then assumption (34) is satisfied for column \( k \) of matrix \( Q \), i.e. variable \( x_k \) appears only linearly in the objective function \( f(x) \) of (QP). For such a column, row \( k \) in (25) then reduces to

\[
c_k - \sum_{j=1}^{n} a_{jk} y_j - z_k = 0.
\]

(37)

**Column Singletons**

A column singleton is a column \( i \) of constraint matrix \( A \) that contains only one non-zero entry:

\[
\exists j : a_{ji} \neq 0, a_{ki} = 0 \quad \forall \; k \neq j.
\]

Let us assume that column \( k \) of \( A \) is a column singleton with \( a_{ik} \neq 0 \). Then, according to (37), row \( i \) in (25) reduces to

\[
c_k - a_{ik} y_i - z_k = 0.
\]

Depending on the bounds \( l_k \) and \( u_k \), different bounds for the dual variable \( y_i \) can be derived from the complementarity conditions (29):

<table>
<thead>
<tr>
<th>( l_k )</th>
<th>( u_k )</th>
<th>( a_{ik} )</th>
<th>( z_k )</th>
<th>( y_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( -\infty )</td>
<td>( +\infty )</td>
<td>( \neq 0 )</td>
<td>0</td>
<td>( c_k / a_{ik} )</td>
</tr>
<tr>
<td>( &gt; -\infty )</td>
<td>( +\infty )</td>
<td>&lt; 0</td>
<td>( \geq 0 )</td>
<td>( \geq c_k / a_{ik} )</td>
</tr>
<tr>
<td>( &gt; -\infty )</td>
<td>( +\infty )</td>
<td>&gt; 0</td>
<td>( \geq 0 )</td>
<td>( \leq c_k / a_{ik} )</td>
</tr>
<tr>
<td>( -\infty )</td>
<td>( &lt; +\infty )</td>
<td>&lt; 0</td>
<td>( \leq 0 )</td>
<td>( \leq c_k / a_{ik} )</td>
</tr>
<tr>
<td>( -\infty )</td>
<td>( &lt; +\infty )</td>
<td>&gt; 0</td>
<td>( \leq 0 )</td>
<td>( \geq c_k / a_{ik} )</td>
</tr>
</tbody>
</table>

Table 2: Bounds on dual variable \( y_i \)
Tightening Bounds on Dual Variables and Fixing Primal Variables

Assume that bounds for the dual vector \( y \) in (25) are known:

\[
\underline{y}_i \leq y_i \leq \overline{y}_i, \quad i = 1, \ldots, m.
\] (38)

Here, \( \underline{y}_i \in \mathbb{R} \cup \{-\infty\} \) and \( \overline{y}_i \in \mathbb{R} \cup \{+\infty\} \).

Similarly to the “Bounds on Primal Constraints and Variables” section, implicit lower and upper bounds can also be derived for the dual constraints (25). Using (38) and the definitions \( P_j = \{ i : a_{ij} > 0 \} \) and \( N_j = \{ i : a_{ij} < 0 \} \), we get the following bounds for \( A^T y \):

\[
\underline{c}_j = \sum_{i \in P_j} a_{ij} \underline{y}_i + \sum_{i \in N_j} a_{ij} \overline{y}_i, \quad \overline{c}_j = \sum_{i \in P_j} a_{ij} \overline{y}_i + \sum_{i \in N_j} a_{ij} \underline{y}_i, \quad j = 1, \ldots, n.
\]

Any \( y \) satisfying (38) now also satisfies

\[
\underline{c}_j \leq \sum_{i=1}^m a_{ij} y_i \leq \overline{c}_j.
\] (39)

Assume that variable \( x_k \) has an infinite upper bound, \( u_k = +\infty \) (The case of an infinite lower bound, \( l_k = -\infty \), can be handled similarly). Because of implication (29), equation (37) reduces to

\[
\sum_{j=1}^m a_{jk} y_j \leq c_k.
\] (40)

Comparing the implied bounds \( \underline{c}_k \) and \( \overline{c}_k \) with \( c_k \), several cases can occur:

- \( \underline{c}_k > c_k \) : Problem (QP) is (dual) infeasible.
- \( \overline{c}_k < c_k \) : For all feasible \( y \), \( z_k \) in (37) is strictly positive. Therefore, due to implication (30), \( x_k = l_k \) if \( l_k \) is finite. In this case, variable \( x_k \) is called (strongly) dominated and it can be eliminated from the problem. If \( l_k = -\infty \), then problem (QP) is unbounded.
- \( \overline{c}_k = c_k \) : Constraint (40) is always satisfied. Variable \( x_k \) is called weakly dominated.
- \( \underline{c}_k \leq c_k < \overline{c}_k \) : Variable \( x_k \) cannot be eliminated from the problem (QP).

Provided (QP) has an optimal solution it can be shown (see [7], the extension to the convex QP case is trivial) that in the weakly dominated case above variable \( x_k \) can be fixed at its lower bound if \( l_k > -\infty \), all constraints \( i \in P_k \) in (26) are of type “greater than or
equal” and all constraints $i \in N_{s_k}$ in (25) are of type “less than or equal”. A similar result holds for the case $l_k = -\infty$, $u_k < +\infty$.

In the case $c_k < c_k < \bar{c}_k$, relations (39) and (40) can also be used to tighten bounds on the dual variables if all dual bounds $\bar{y}_i$ and $\underline{y}_i$ are finite for $i \in P_{*j}$ and $l \in N_{*j}$, respectively:

$$\forall j \in P_{*k} : \quad c_k + a_{jk}(y_j - \bar{y}_j) \leq \sum_{i=1}^{m} a_{ik}y_i \leq c_k \quad \Rightarrow \quad y_j \leq \bar{y}_j + \frac{c_k - \bar{c}_k}{a_{jk}} =: \bar{y}_j',$$

$$\forall j \in N_{*k} : \quad c_k + a_{jk}(y_j - \bar{y}_j) \leq \sum_{i=1}^{m} a_{ik}y_i \leq c_k \quad \Rightarrow \quad y_j \geq \bar{y}_j + \frac{c_k - \bar{c}_k}{a_{jk}} =: \underline{y}_j'.$$

These new bounds can be redundant, tightening or contradictory to the old bounds. In the latter case, the problem is dual infeasible.

In the case $c_k = \bar{c}_k$, it follows again from (39) and (40) that the variables $y_i$, $l \in P_{*j} \cup N_{*j}$, can be fixed at their respective bounds, $y_i = \bar{y}_i$ for $l \in P_{*j}$ and $y_i = \underline{y}_i$ for $l \in N_{*j}$.

Note that new dual bounds can also be obtained if exactly one of the bounds $\bar{y}_i$ with $l \in P_{*j}$ or $\underline{y}_i$ with $l \in N_{*j}$ is infinite.

If sparse_quadratic_prog is called with a presolve level greater than 2 then all techniques described in this section are applied to problem (QP).

**Elimination of Column Singletons**

Let us assume that column $k$ of constraint matrix $A$ is a column singleton with $a_{ik} \neq 0$ and that assumption (34) is satisfied for column $k$ of matrix $Q$, i.e. variable $x_k$ appears only linearly in the objective function $f(x)$ of (QP).

Depending on the bounds on $x_k$, we will consider two different cases:

a) $x_k$ is free ($l_k = -\infty \land u_k = +\infty$)

Because of table 2 in the last section, row $k$ of (25) reduces to

$$y_i = c_k / a_{ik}.$$

Assume first that $c_k \neq 0$ so that $y_i \neq 0$. Then, according to (32) and (33), row $i$ of (26) is actually an equality:

$$\sum_{j=1}^{n} a_{ij}x_j = \bar{b}_i \quad \text{if} \quad y_i > 0, \quad \sum_{j=1}^{n} a_{ij}x_j = \underline{b}_i \quad \text{if} \quad y_i < 0.$$
Since $x_k$ is free and appears only in row $i$ of $A$, it can be eliminated from problem (QP):

$$x_k = (\bar{b}_i - \sum_{j \neq k} a_{ij} x_j) / a_{ik} \text{ if } y_i > 0, \quad x_k = (\bar{b}_i - \sum_{j \neq k} a_{ij} x_j) / a_{ik} \text{ if } y_i < 0.$$ 

The updated vector $c$ is then defined by

$$c'_j = c_j - \frac{a_{ij}}{a_{ik}} c_k \forall \ j \neq k, \ c'_k = 0.$$ 

If $c_k = 0$, then the value of the objective function $f(x)$ is not influenced by the value of $x_k$. Therefore any feasible $x_k$ can also be part of an optimal solution. Since $x_k$ is free and appears only in the $i$-th constraint of (26), both $x_k$ and constraint $i$ can be eliminated without further adjustments to problem (QP). If an optimal solution $(x_1^*, ..., x_{k-1}^*, x_{k+1}^*, ..., x_n^*)$ for the reduced problem is found, then every $x_k$ satisfying

$$\bar{b}_i - \sum_{j \neq k} a_{ij} x_j^* + a_{ik} x_k \leq \bar{b}_i$$

completes an optimal solution $x^*$ for (QP).

b) $x_k$ is bounded ($l_k \not= -\infty \lor u_k \not= +\infty$)

Set $P_\star = \{j : a_{ij} > 0\}$, $N_\star = \{j : a_{ij} < 0\}$ and

$$r_i = \sum_{j \in P_\star \setminus \{k\}} a_{ij} l_j + \sum_{j \in N_\star \setminus \{k\}} a_{ij} u_j, \quad r_u = \sum_{j \in P_\star \setminus \{k\}} a_{ij} u_j + \sum_{j \in N_\star \setminus \{k\}} a_{ij} l_j.$$ 

It follows from (26) and (27) that the following implied bounds $l'_k$ and $u'_k$ can be derived for $x_k$:

$$l'_k = \begin{cases} (\bar{b}_i - r_u) / a_{ik} & \text{if } a_{ik} > 0 \\ (\bar{b}_i - r_i) / a_{ik} & \text{if } a_{ik} < 0 \end{cases}$$

$$u'_k = \begin{cases} (\bar{b}_i - r_i) / a_{ik} & \text{if } a_{ik} > 0 \\ (\bar{b}_i - r_u) / a_{ik} & \text{if } a_{ik} < 0 \end{cases}.$$ 

If $l_k \leq l'_k \leq u'_k \leq u_k$, then variable $x_k$ is implicitly free so that the techniques described under a) can be applied.

Function `sparse_quadratic_prog` implements the elimination of column singletons for presolve levels greater than 3.
Removal of Doubleton Equality Rows

A doubleton equality row is a constraint row of the form

$$a_{ij}x_j + a_{ik}x_k = b_j, \quad a_{ij} \neq 0, \quad a_{ik} \neq 0.$$  

By setting

$$x_j = \frac{b_j}{a_{ij}} - \frac{a_{ik}}{a_{ij}} x_k,$$  

row $i$ can be eliminated from the problem. Denoting the reduced constraint matrix by $A'$ and the reduced upper and lower bounds by $b'$ and $\overline{b}'$, respectively, columns $a_{i,j}'$ and $a_{i,k}'$ are replaced by the aggregate column

$$a_{i,k}' - \frac{a_{ik}}{a_{ij}} a_{i,j}'$$

while $b'$ and $\overline{b}'$ are replaced by

$$b' - \frac{b_j}{a_{ij}} a_{i,j}' \quad \text{and} \quad \overline{b}' - \frac{b_j}{a_{ij}} a_{i,j}'$$

respectively. Furthermore, variable $x_j$ gets implicitly free by using the bounds $l_j$ and $u_j$ and relation (41) to adjust the bounds for $x_k$.

Note that the generation of aggregate columns might cause fill-in in $A$.

Function `sparse_quadratic_prog` removes doubleton equality rows from problem (QP) for presolve levels greater than 4.

Column Aggregation

Under duplicate columns we identify columns in the constraint matrix $A$ that are linearly dependent:

$$a_{i,j} = \alpha a_{i,k}, \quad \alpha \in \mathbb{R}.$$  

If only reductions on the constraints (26) are considered, then duplicate columns can be aggregated and the corresponding variables can be replaced by a new variable. For example, if the aggregate contains the two columns $a_{i,j}$ and $a_{i,k}$, then

$$a_{i,k}x_k + a_{i,j}x_j = a_{i,k} (x_k + \alpha x_j),$$
and therefore, variables $x_j$ and $x_k$ can be replaced by

$$x'_k = x_k + \alpha x_j$$

with bounds

$$l'_k = \begin{cases} l_k + \alpha l_j & \text{if } \alpha > 0 \\ l_k + \alpha u_k & \text{if } \alpha < 0 \end{cases}$$

and

$$u'_k = \begin{cases} u_k + \alpha u_j & \text{if } \alpha > 0 \\ u_k + \alpha l_k & \text{if } \alpha < 0 \end{cases}$$

In general, after aggregation, the columns of constraint matrix $A$ are partitioned into columns belonging to aggregate variables and columns belonging to standard variables of the form (27). The techniques in “Bounds on Primal Constraints and Variables” can now also be applied to this form of constraints to detect redundant and forcing constraints and to tighten bounds on standard and aggregated variables.

We want to show by an example, how the values of the variables in an aggregated column can be determined if the aggregated column is part of a forcing constraint.

Assume in the following that constraint (26) contains one aggregate with three variables $x_j$, $x_k$ and $x_l$ and that for the index set $J := \{j, k, l\}$ assumption (34) is satisfied. Using the notation $M := \{1, \ldots, n\} \setminus J$, problem (QP) then appears as follows:

$$\begin{align*}
\min_{\mathbf{x} \in \mathbb{R}^n} & \quad f(x) = c_M^T x_M + c_j^T x_j + \frac{1}{2} x_M^T Q_{M,M} x_M \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad
and

\[
\begin{align*}
\text{min} & \quad c_j x_j = c_j x_j + c_k x_k + c_i x_i \\
\quad (\text{LP}) & \quad x_j + \alpha_k x_k + \alpha_i x_i = u_j' \\
& \quad l_j \leq x_j \leq u_j,
\end{align*}
\]

Problem (LP) is an easy-to-solve linear subproblem of (QP). Its solution \((x_j^*, x_k^*, x_i^*)\) can be set back into (QP) in order to fix variables \(x_j\).

Aggregates that contain only two variables can be used directly for reductions on the entire (QP) problem. Assume again that for the doubleton columns \(k\) and \(j\) condition (34) is satisfied so that variables \(x_k\) and \(x_j\) appear only linearly in the objective function \(f(x)\). Furthermore, assume that relations (43) - (45) are valid for \(x_k\) and \(x_j\).

Depending on the relationship between the linear coefficients \(c_k\), \(c_j\) in \(f(x)\), two different cases must be considered:

- **Case 1**: \(c_j = \alpha c_k\)

  Variables \(x_k\) and \(x_j\) can be replaced by \(x_k'\) in (44) with bounds as in (45).

- **Case 2**: \(c_j \neq \alpha c_k\)

  Denoting a solution to (25) by \((x^*, y^*, z^*)\), it follows from (25) and (34) that

\[
z_i^* = c_i - a_{ik} y^*, \quad i = j, k.
\]

(46)

Assume that \(c_j > \alpha c_k\) (The case \(c_j < \alpha c_k\) is similar).

If \((l_k > -\infty \wedge u_k = +\infty \wedge \alpha \geq 0)\) or \((l_k = -\infty \wedge u_k < +\infty \wedge \alpha \leq 0)\) then (29), (42), (46) result in

\[
z_j^* = c_j - a_{kj} y^* > \alpha (c_k - a_{ik} y^*) = \alpha z_k^* \geq 0,
\]

so that according to (29) column \(k\) dominates column \(j\) if \(l_j > -\infty\):

\[
x_j^* = l_j.
\]

If \(l_j = -\infty\) then problem (QP) is dual infeasible.

Function `sparse_quadratic_prog` eliminates aggregate columns for the highest presolve level 6.
Numerical Results

We have tested `sparse_quadratic_prog` on a series of convex QP test problems of all sizes from the repository compiled by Maros and Mészáros (see [10]). For the computations we used the default parameter values defined for `sparse_quadratic_prog` with two exceptions: The presolve level was set to the highest value 6 and the maximum iteration number was extended from 200 to 1000. The tests were conducted on an Intel® IA-32-platform with OS Windows XP and the Microsoft® 32-bit C/C++ Optimizing Compiler Version 16.00.30319.01 shipped with Microsoft Visual Studio 2010. The test results are summarized in Table 3: Solving Sparse Convex Quadratic Programming Problems. Here, $m$ and $n$ denote the number of rows and columns of constraint matrix $A$, respectively, $\text{nz}(A)$ is the number of nonzeros in $A$ and $\text{nz}(Q)$ is the number of nonzeros in the lower triangular part (including the diagonal) of $Q$. Under Iterations, the number of predictor-corrector steps needed to find the optimal solution is summarized. Time is the total solution time (excluding the time required to read in the QPS file) and is measured in CPU seconds.

Note that the large computation times for boyd2 and cont-300 are not caused by the interior-point method itself but by the minimum degree ordering of the augmented system matrix which is done just before the optimization starts. Possible improvements here are currently under investigation.
<table>
<thead>
<tr>
<th>Problem</th>
<th>(m)</th>
<th>(n)</th>
<th>(nz(A))</th>
<th>(nz(Q))</th>
<th>Iterations</th>
<th>Time</th>
<th>Objective Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>aug2dcqp</td>
<td>10000</td>
<td>2020</td>
<td>40000</td>
<td>20200</td>
<td>12</td>
<td>6.83</td>
<td>6.4981347400e+06</td>
</tr>
<tr>
<td>aug2dqp</td>
<td>10000</td>
<td>2020</td>
<td>40000</td>
<td>19800</td>
<td>14</td>
<td>7.16</td>
<td>6.2370120267e+06</td>
</tr>
<tr>
<td>aug3dcqp</td>
<td>1000</td>
<td>3873</td>
<td>6546</td>
<td>3837</td>
<td>12</td>
<td>0.48</td>
<td>9.9336237916e+02</td>
</tr>
<tr>
<td>aug3dqp</td>
<td>1000</td>
<td>3873</td>
<td>6546</td>
<td>2673</td>
<td>14</td>
<td>0.52</td>
<td>6.7523806159e+02</td>
</tr>
<tr>
<td>boyd1</td>
<td>18</td>
<td>9326</td>
<td>558985</td>
<td>93261</td>
<td>22</td>
<td>18.63</td>
<td>-6.1735219504e+07</td>
</tr>
<tr>
<td>boyd2</td>
<td>18653</td>
<td>9326</td>
<td>423784</td>
<td>2</td>
<td>36</td>
<td>8827.0</td>
<td>2.1256767237e+01</td>
</tr>
<tr>
<td>cont-050</td>
<td>2401</td>
<td>2597</td>
<td>12005</td>
<td>2597</td>
<td>8</td>
<td>1.08</td>
<td>-4.5638507724e+00</td>
</tr>
<tr>
<td>cont-100</td>
<td>9801</td>
<td>1019</td>
<td>49005</td>
<td>10197</td>
<td>9</td>
<td>13.91</td>
<td>-4.6443978608e+00</td>
</tr>
<tr>
<td>cont-101</td>
<td>10098</td>
<td>1019</td>
<td>49599</td>
<td>2700</td>
<td>9</td>
<td>15.14</td>
<td>1.9552732481e-01</td>
</tr>
<tr>
<td>cont-200</td>
<td>39601</td>
<td>4039</td>
<td>198005</td>
<td>40397</td>
<td>9</td>
<td>372.63</td>
<td>-4.6848759035e+00</td>
</tr>
<tr>
<td>cont-201</td>
<td>40198</td>
<td>4039</td>
<td>199199</td>
<td>10400</td>
<td>9</td>
<td>400.63</td>
<td>1.9248337442e-01</td>
</tr>
<tr>
<td>cont-300</td>
<td>90298</td>
<td>9059</td>
<td>448799</td>
<td>23100</td>
<td>10</td>
<td>3538.2</td>
<td>1.9151228592e-01</td>
</tr>
<tr>
<td>cvxqp1_l</td>
<td>5000</td>
<td>1000</td>
<td>14998</td>
<td>39984</td>
<td>7</td>
<td>315.81</td>
<td>1.0870483090e+08</td>
</tr>
<tr>
<td>cvxqp1_m</td>
<td>500</td>
<td>1000</td>
<td>1498</td>
<td>3984</td>
<td>10</td>
<td>2.83</td>
<td>1.0875115685e+06</td>
</tr>
<tr>
<td>cvxqp2_l</td>
<td>2500</td>
<td>1000</td>
<td>7499</td>
<td>39984</td>
<td>8</td>
<td>145.70</td>
<td>8.1842460378e+07</td>
</tr>
<tr>
<td>cvxqp2_m</td>
<td>250</td>
<td>1000</td>
<td>749</td>
<td>3984</td>
<td>8</td>
<td>0.45</td>
<td>8.2015543332e+05</td>
</tr>
<tr>
<td>cvxqp3_l</td>
<td>7500</td>
<td>1000</td>
<td>22497</td>
<td>39984</td>
<td>7</td>
<td>343.09</td>
<td>1.1571121170e+08</td>
</tr>
<tr>
<td>cvxqp3_m</td>
<td>750</td>
<td>1000</td>
<td>2247</td>
<td>3984</td>
<td>9</td>
<td>1.25</td>
<td>1.3628287429e+06</td>
</tr>
<tr>
<td>dpklo1</td>
<td>77</td>
<td>133</td>
<td>1575</td>
<td>77</td>
<td>4</td>
<td>0.11</td>
<td>3.7009621711e-01</td>
</tr>
<tr>
<td>dtoc3</td>
<td>9998</td>
<td>1499</td>
<td>34993</td>
<td>14997</td>
<td>13</td>
<td>4.58</td>
<td>2.3526258052e+02</td>
</tr>
<tr>
<td>gouldqp2</td>
<td>349</td>
<td>699</td>
<td>1047</td>
<td>697</td>
<td>13</td>
<td>0.19</td>
<td>1.8427450348e-04</td>
</tr>
<tr>
<td>gouldqp3</td>
<td>349</td>
<td>699</td>
<td>1047</td>
<td>1395</td>
<td>9</td>
<td>0.22</td>
<td>2.0627839716e+00</td>
</tr>
<tr>
<td>laser</td>
<td>1000</td>
<td>1002</td>
<td>3000</td>
<td>4002</td>
<td>31</td>
<td>4.11</td>
<td>2.4096013456e+06</td>
</tr>
<tr>
<td>lotschd</td>
<td>7</td>
<td>12</td>
<td>54</td>
<td>6</td>
<td>10</td>
<td>0.02</td>
<td>2.3984158915e+03</td>
</tr>
<tr>
<td>mosarqp1</td>
<td>700</td>
<td>2500</td>
<td>3422</td>
<td>2545</td>
<td>9</td>
<td>1.20</td>
<td>-9.5287543932e+02</td>
</tr>
<tr>
<td>mosarqp2</td>
<td>600</td>
<td>900</td>
<td>2930</td>
<td>945</td>
<td>10</td>
<td>0.88</td>
<td>-1.5974826056e+03</td>
</tr>
<tr>
<td>powell20</td>
<td>10000</td>
<td>1000</td>
<td>20000</td>
<td>10000</td>
<td>120</td>
<td>57.47</td>
<td>5.2089590625e+10</td>
</tr>
<tr>
<td>q25fv47</td>
<td>820</td>
<td>1571</td>
<td>10400</td>
<td>59499</td>
<td>23</td>
<td>6.19</td>
<td>1.3744447892e+07</td>
</tr>
<tr>
<td>qadlittl</td>
<td>56</td>
<td>97</td>
<td>383</td>
<td>87</td>
<td>9</td>
<td>0.02</td>
<td>4.8031885955e+05</td>
</tr>
<tr>
<td>qafiro</td>
<td>27</td>
<td>32</td>
<td>83</td>
<td>6</td>
<td>7</td>
<td>0.02</td>
<td>-1.5907817909e+00</td>
</tr>
<tr>
<td>qbandm</td>
<td>305</td>
<td>472</td>
<td>2494</td>
<td>41</td>
<td>11</td>
<td>0.11</td>
<td>1.6352342059e+04</td>
</tr>
</tbody>
</table>

Table 3: Solving Sparse Convex Quadratic Programming Problems
<table>
<thead>
<tr>
<th>Problem</th>
<th>m</th>
<th>n</th>
<th>nz(A)</th>
<th>nz(Q)</th>
<th>Iterations</th>
<th>Time</th>
<th>Objective Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>qbeaconf</td>
<td>173</td>
<td>262</td>
<td>3375</td>
<td>27</td>
<td>12</td>
<td>0.03</td>
<td>1.6471206018e+05</td>
</tr>
<tr>
<td>qbore3d</td>
<td>233</td>
<td>315</td>
<td>1429</td>
<td>78</td>
<td>13</td>
<td>0.05</td>
<td>3.1002008020e+03</td>
</tr>
<tr>
<td>qbrandy</td>
<td>220</td>
<td>249</td>
<td>2148</td>
<td>65</td>
<td>13</td>
<td>0.08</td>
<td>2.8375114858e+04</td>
</tr>
<tr>
<td>qcapri</td>
<td>271</td>
<td>353</td>
<td>1767</td>
<td>894</td>
<td>17</td>
<td>0.16</td>
<td>6.6793294005e+07</td>
</tr>
<tr>
<td>qe226</td>
<td>223</td>
<td>282</td>
<td>2578</td>
<td>964</td>
<td>12</td>
<td>0.14</td>
<td>2.1265343417e+02</td>
</tr>
<tr>
<td>qetamacr</td>
<td>400</td>
<td>688</td>
<td>2409</td>
<td>4447</td>
<td>17</td>
<td>0.48</td>
<td>8.6760311985e+04</td>
</tr>
<tr>
<td>qffiff80</td>
<td>524</td>
<td>854</td>
<td>6227</td>
<td>1916</td>
<td>15</td>
<td>0.53</td>
<td>8.7314749076e+05</td>
</tr>
<tr>
<td>qforplan</td>
<td>161</td>
<td>421</td>
<td>4563</td>
<td>582</td>
<td>19</td>
<td>0.34</td>
<td>7.4566315148e+09</td>
</tr>
<tr>
<td>qgfrdxpn</td>
<td>616</td>
<td>1092</td>
<td>2377</td>
<td>162</td>
<td>12</td>
<td>0.20</td>
<td>1.0079058863e+11</td>
</tr>
<tr>
<td>qgrow15</td>
<td>300</td>
<td>645</td>
<td>5620</td>
<td>500</td>
<td>15</td>
<td>0.44</td>
<td>-1.0169364047e+08</td>
</tr>
<tr>
<td>qgrow22</td>
<td>440</td>
<td>946</td>
<td>8252</td>
<td>852</td>
<td>20</td>
<td>0.98</td>
<td>-1.4962895347e+08</td>
</tr>
<tr>
<td>qgrow7</td>
<td>140</td>
<td>301</td>
<td>2612</td>
<td>357</td>
<td>14</td>
<td>0.17</td>
<td>-4.2798713873e+07</td>
</tr>
<tr>
<td>qisrael</td>
<td>174</td>
<td>142</td>
<td>2269</td>
<td>698</td>
<td>17</td>
<td>0.11</td>
<td>2.5347837791e+07</td>
</tr>
<tr>
<td>qpcblend</td>
<td>74</td>
<td>83</td>
<td>491</td>
<td>83</td>
<td>15</td>
<td>0.03</td>
<td>-7.8425428230e-03</td>
</tr>
<tr>
<td>qpcboei1</td>
<td>351</td>
<td>384</td>
<td>3485</td>
<td>384</td>
<td>13</td>
<td>0.14</td>
<td>1.1503914079e+07</td>
</tr>
<tr>
<td>qpcboei2</td>
<td>166</td>
<td>143</td>
<td>1196</td>
<td>143</td>
<td>25</td>
<td>0.08</td>
<td>8.1719622443e+06</td>
</tr>
<tr>
<td>qpcstair</td>
<td>356</td>
<td>467</td>
<td>3856</td>
<td>467</td>
<td>14</td>
<td>0.30</td>
<td>6.2043874821e+06</td>
</tr>
<tr>
<td>qpilotno</td>
<td>975</td>
<td>2172</td>
<td>13057</td>
<td>485</td>
<td>14</td>
<td>1.92</td>
<td>4.7285869049e+06</td>
</tr>
<tr>
<td>qrecipe</td>
<td>91</td>
<td>180</td>
<td>663</td>
<td>50</td>
<td>12</td>
<td>0.02</td>
<td>-2.6661599957e+02</td>
</tr>
<tr>
<td>qsc205</td>
<td>205</td>
<td>203</td>
<td>551</td>
<td>21</td>
<td>12</td>
<td>0.05</td>
<td>-5.8139482875e-03</td>
</tr>
<tr>
<td>qscagr25</td>
<td>471</td>
<td>500</td>
<td>1554</td>
<td>128</td>
<td>11</td>
<td>0.11</td>
<td>2.0173793974e+08</td>
</tr>
<tr>
<td>qscagr7</td>
<td>129</td>
<td>140</td>
<td>420</td>
<td>25</td>
<td>10</td>
<td>0.02</td>
<td>2.6865948619e+07</td>
</tr>
<tr>
<td>qscfxm1</td>
<td>330</td>
<td>457</td>
<td>2589</td>
<td>733</td>
<td>21</td>
<td>0.25</td>
<td>1.6882691640e+07</td>
</tr>
<tr>
<td>qscfxm2</td>
<td>660</td>
<td>914</td>
<td>5183</td>
<td>1131</td>
<td>23</td>
<td>0.64</td>
<td>2.7776161582e+07</td>
</tr>
<tr>
<td>qscfxm3</td>
<td>990</td>
<td>1371</td>
<td>7777</td>
<td>1221</td>
<td>21</td>
<td>0.92</td>
<td>3.0816354480e+07</td>
</tr>
<tr>
<td>qscorpio</td>
<td>388</td>
<td>358</td>
<td>1426</td>
<td>40</td>
<td>9</td>
<td>0.06</td>
<td>1.8805095547e+03</td>
</tr>
<tr>
<td>qscrs8</td>
<td>490</td>
<td>1169</td>
<td>3182</td>
<td>121</td>
<td>13</td>
<td>0.25</td>
<td>9.0456003747e+02</td>
</tr>
<tr>
<td>qscsd1</td>
<td>77</td>
<td>760</td>
<td>2388</td>
<td>745</td>
<td>8</td>
<td>0.09</td>
<td>8.6666666772e+00</td>
</tr>
<tr>
<td>qscsd6</td>
<td>147</td>
<td>1350</td>
<td>4316</td>
<td>1404</td>
<td>9</td>
<td>0.25</td>
<td>5.0808263940e+01</td>
</tr>
<tr>
<td>qscsd8</td>
<td>397</td>
<td>2750</td>
<td>8584</td>
<td>2510</td>
<td>9</td>
<td>0.73</td>
<td>9.4076357842e+02</td>
</tr>
<tr>
<td>qstap1</td>
<td>300</td>
<td>480</td>
<td>1692</td>
<td>153</td>
<td>12</td>
<td>0.14</td>
<td>1.4158611200e+03</td>
</tr>
<tr>
<td>qstap2</td>
<td>1090</td>
<td>1880</td>
<td>6714</td>
<td>777</td>
<td>10</td>
<td>0.59</td>
<td>1.7350265642e+03</td>
</tr>
<tr>
<td>Problem</td>
<td>m</td>
<td>n</td>
<td>nz(A)</td>
<td>nz(Q)</td>
<td>Iterations</td>
<td>Time</td>
<td>Objective Value</td>
</tr>
<tr>
<td>----------</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>------------</td>
<td>-------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>qsctap3</td>
<td>1480</td>
<td>2480</td>
<td>8874</td>
<td>1047</td>
<td>11</td>
<td>0.95</td>
<td>1.4387546871e+03</td>
</tr>
<tr>
<td>qseba</td>
<td>515</td>
<td>1028</td>
<td>4352</td>
<td>646</td>
<td>19</td>
<td>0.08</td>
<td>8.1481800536e+07</td>
</tr>
<tr>
<td>qshare1b</td>
<td>117</td>
<td>225</td>
<td>1151</td>
<td>39</td>
<td>14</td>
<td>0.06</td>
<td>7.2007831958e+05</td>
</tr>
<tr>
<td>qshare2b</td>
<td>96</td>
<td>79</td>
<td>694</td>
<td>55</td>
<td>12</td>
<td>0.03</td>
<td>1.1703691723e+04</td>
</tr>
<tr>
<td>qshell</td>
<td>536</td>
<td>1775</td>
<td>3556</td>
<td>34790</td>
<td>13</td>
<td>0.42</td>
<td>1.5726370020e+12</td>
</tr>
<tr>
<td>qship04l</td>
<td>402</td>
<td>2118</td>
<td>6332</td>
<td>56</td>
<td>8</td>
<td>0.34</td>
<td>2.4200156089e+06</td>
</tr>
<tr>
<td>qship04s</td>
<td>402</td>
<td>1458</td>
<td>4352</td>
<td>56</td>
<td>9</td>
<td>0.19</td>
<td>2.4249936790e+06</td>
</tr>
<tr>
<td>qship08l</td>
<td>778</td>
<td>4283</td>
<td>12802</td>
<td>34965</td>
<td>10</td>
<td>2.88</td>
<td>2.3760406307e+06</td>
</tr>
<tr>
<td>qship08s</td>
<td>778</td>
<td>2387</td>
<td>7114</td>
<td>11677</td>
<td>9</td>
<td>0.52</td>
<td>2.3857288949e+06</td>
</tr>
<tr>
<td>qship12l</td>
<td>1151</td>
<td>5427</td>
<td>16170</td>
<td>62228</td>
<td>9</td>
<td>3.91</td>
<td>3.0188766047e+06</td>
</tr>
<tr>
<td>qship12s</td>
<td>1151</td>
<td>2763</td>
<td>8178</td>
<td>17403</td>
<td>9</td>
<td>0.61</td>
<td>3.0569622859e+06</td>
</tr>
<tr>
<td>qsierra</td>
<td>1227</td>
<td>2036</td>
<td>7302</td>
<td>183</td>
<td>13</td>
<td>0.69</td>
<td>2.3750458510e+07</td>
</tr>
<tr>
<td>qstair</td>
<td>356</td>
<td>467</td>
<td>3856</td>
<td>1018</td>
<td>15</td>
<td>0.25</td>
<td>7.9854527490e+06</td>
</tr>
<tr>
<td>qstandat</td>
<td>359</td>
<td>1075</td>
<td>3031</td>
<td>804</td>
<td>10</td>
<td>0.14</td>
<td>6.4118385480e+03</td>
</tr>
<tr>
<td>stdat1</td>
<td>3999</td>
<td>2001</td>
<td>9997</td>
<td>2000</td>
<td>61</td>
<td>4.22</td>
<td>-2.8525533676e+03</td>
</tr>
<tr>
<td>stdat2</td>
<td>3999</td>
<td>2001</td>
<td>9997</td>
<td>2000</td>
<td>247</td>
<td>11.33</td>
<td>-3.2626655151e+01</td>
</tr>
<tr>
<td>stdat3</td>
<td>7999</td>
<td>4001</td>
<td>19997</td>
<td>4000</td>
<td>861</td>
<td>275.67</td>
<td>-3.5779466838e+01</td>
</tr>
<tr>
<td>stcqp1</td>
<td>2052</td>
<td>4097</td>
<td>13338</td>
<td>26603</td>
<td>9</td>
<td>0.20</td>
<td>1.5514356000e+05</td>
</tr>
<tr>
<td>stcqp2</td>
<td>2052</td>
<td>4097</td>
<td>13338</td>
<td>26603</td>
<td>9</td>
<td>0.34</td>
<td>2.2327327237e+04</td>
</tr>
<tr>
<td>ubh1</td>
<td>12000</td>
<td>1800</td>
<td>48000</td>
<td>6003</td>
<td>7</td>
<td>1.33</td>
<td>1.1160008215e+00</td>
</tr>
</tbody>
</table>
Conclusion

In this article, we described some important mathematical and numerical aspects of the sparse convex quadratic programming function \texttt{sparse\_quadratic\_prog} implemented in IMSL® C Numerical Library 8.0. Based on Jacek Gondzio’s code HOPDM, it uses an infeasible primal-dual interior-point method, today the de-facto standard for solving large sparse linear and convex quadratic programming problems, for the solution of the resulting Karush-Kuhn-Tucker conditions. As indicated by the numerical results, the use of weighted multiple centrality correctors helps to reduce the number of iteration steps required to solve even large problems. The code is also very robust — the dense QP solver \texttt{quadratic\_prog}, currently in our library, is only able to solve a subset of the smaller problems mentioned in Table 3. By focusing on the augmented system approach with primal and dual regularizations, in every step of the interior-point method the search directions can be found accurately via a Cholesky-like method. Finally, the use of advanced presolving techniques can lead to significant reductions in the problem size which in turn helps to save computation time.

About the IMSL Libraries

The IMSL Numerical Libraries are a comprehensive set of mathematical and statistical functions that programmers can embed into their software applications. The libraries save development time by providing pre-written mathematical and statistical algorithms that can be embedded into C, C# for .NET, Java and Fortran applications, enhancing return on investment and programmer productivity. The IMSL Libraries can also be used from Python. Beyond choice of programming language, the IMSL Libraries are supported across a wide range of hardware and operating system environments including Windows, Linux, Apple and many UNIX platforms. Go to http://www.roguewave.com/products/imsl-numerical-libraries.aspx to view our product information and links to datasheets, videos, tutorials, evaluation options and more.

About Rogue Wave Software

Rogue Wave Software, Inc. is the largest independent provider of cross-platform software development tools and embedded components for the next generation of HPC applications. Rogue Wave marries High Performance Computing with High Productivity Computing to enable developers to harness the power of parallel applications and multicore computing. Rogue Wave products reduce the complexity of prototyping, developing, debugging, and optimizing multiprocessor and data-intensive applications. Rogue Wave customers are industry leaders in the Global 2000, ISVs, OEMs, government laboratories and research institutions that leverage computationally-complex and data-intensive applications to enable innovation and outperform competitors. For more information, visit http://www.roguewave.com.
References


