On a conjugate directions method for solving strictly convex QP problem

Andrzej Stachurski

Abstract Problem of solving the strictly convex, quadratic programming problem is studied. The idea of conjugate directions is used. First we assume that we know the set of directions conjugate with respect to the hessian of the goal function. We apply \( n \) simultaneous directional minimizations along these conjugate directions starting from the same point followed by the addition of the directional corrections. Theorem justifying that the algorithm finds the global minimum of the quadratic goal function is proved. The way of effective construction of the required set of conjugate directions is presented. We start with a vector with zero value entries except the first one. At each step new vector conjugate to the previously generated is constructed whose number of nonzero entries is larger by one than in its predecessor. Conjugate directions obtained by means of the above construction procedure with appropriately selected parameters form an upper triangular matrix which in exact computations is the Cholesky factor of the inverse of the hessian matrix. Computational cost of calculating the inverse factorization is comparable with the cost of the Cholesky factorization of the original second derivative matrix. Calculation of those vectors involves exclusively matrix/vector multiplication and finding an inverse of a diagonal matrix. Some preliminary computational results on some test problems are reported. In the test problems all symmetric, positive definite matrices with dimensions from \( 14 \times 14 \) to \( 2000 \times 2000 \) from the repository of the Florida University were used as the hessians.

Keywords Conjugate directions · Quadratic programming problems · Distributed calculations
Mathematics Subject Classification 90C20 · 65K05 · 90-08 · 65F05

1 Introduction

In the paper we consider problem of solving the following unconstrained optimization problem

\[ \min_{x \in \mathbb{R}^n} f(x) = -b^T x + \frac{1}{2} x^T G x. \]  

Equation (3) corresponds to the first order necessary optimality conditions for problem (1)

\[ \nabla f(x) = 0 \iff -b + Gx = 0, \]  

where matrix \( G \) is symmetric and positive definite.

The problem is equivalent to the problem of finding the solution of the set of linear equations

\[ Gx = b. \]  

The equivalence of (3) and (1) is easily verified.

There are two main approaches to solve the quadratic problem (1) or the set of linear equations (3). In the first, a sequence of consecutive directional minimizations along conjugate directions (first of all conjugate gradients) is carried out. In the second approach, one first finds the triangular Cholesky decomposition of matrix \( G \) and next two sets of linear equations with triangular matrices are solved.

The notion of the conjugate directions plays very important role in the solution methods of the QP problem (1) and general unconstrained nonlinear minimization problems. Powell (1964) has proposed the conjugate directions method without calculating the derivatives. The consecutive direction was generated as the difference between the minima found on two parallel straight lines. The most popular are the conjugate gradient methods. The first one was developed by Hestenes and Stiefel (1952), who proposed it for solving symmetric, positive-definite linear algebraic systems. In 1964 the method has been extended to nonlinear problems by Fletcher and Reeves (1964). The procedure is recursive

\[ x^{k+1} = x^k + \alpha^k d^k, \]

where \( d^0 = -\nabla f(x^0) \) and every consecutive search direction is the linear combination of the previous one and the Cauchy steepest descent direction

\[ d^{k+1} = -\nabla f(x^{k+1}) + \beta^k d^k. \]

It means that starting from an initial guess \( x^0 \) (with corresponding residual \( r^0 = \nabla f(x^0) = Gx^0 - b \)), this method generates a sequence of iterates \( \{x^k\} \) that minimize \( f \) over the Krylov subspaces \( x^0 + \text{span}(r^0, Gr^0, \ldots, G^{k-1}r^0) \). Since the publication of the Hestenes and Stiefel (1952), conjugate gradient methods have become an area of intensive research and large number of variants of conjugate gradient algorithms have been designed. Some well-known choices for \( \beta^k \) are Fletcher and Reeves (1964),
Polak and Ribiére (1969), Polyak (1969), Dai and Yuan (1999) and Hestenes and Stiefel (1952):

$$
\beta_{FR}^k = \frac{\|g^{k+1}\|}{\|g^k\|}, \quad \beta_{PRP}^k = \frac{(g^{k+1})^T r_k}{\|g^k\|}, \quad \beta_{DY}^k = \frac{\|g^{k+1}\|^2}{(d^k)^T r_k}, \quad \beta_{HS}^k = \frac{(g^{k+1})^T r_k}{(d^k)^T r_k}
$$

respectively, where $g^k = \nabla f(x^k)$, $r^k = g^{k+1} - g^k$ and $\|\cdot\|$ means the Euclidean norm. Surveys on the developments in that area one may find for instance in Hager and Zhang (2005), Stachurski (2012) and in Dai (2011).

Standard algebraic approach to solve either the set of linear equations (1) or the quadratic problem (3) is two-phase. In phase one, the triangular Cholesky decomposition (published post-mortem by Benoit (1924)) of the hessian matrix is found, i.e., the lower triangular matrix $L$ such that $G = LL^T$. In phase two, two sets of linear equations with triangular matrices are consecutively solved using the forward substitution followed by the backward substitution in order to find the solution of (1)

$$
Ly = b \quad \text{and} \quad L^T x = y. \tag{4}
$$

A closely related approach is to express the hessian as $G = LDL^T$, where $L$ is lower triangular with ones on the main diagonal and $D$ is a diagonal matrix. Similarly as in formula (4), after finding the decomposition, two sets with triangular matrices and one with a diagonal matrix are solved

$$
Ly = b \quad \text{and} \quad Dz = y \quad \text{and} \quad L^T x = z. \tag{5}
$$

A survey of direct methods for sparse linear systems one may find in the Davis et al. (2016) technical report. Problems of type (1,3) are a kind of inverse problems and have many applications. An inverse problem in science is the process of calculating from a set of observations the unknown input values: for instance calculating an image in computer tomography, deblurring the digital picture, reconstructing the source in acoustics or calculating the density of the Earth from measurements of its gravity field (for more information see Wikipedia (2017)). Many of them involve large-scale problems with symmetric, positive definite matrices and multiple right-hand sides.

We observe recently a renewed interest in large, sparse, unconstrained quadratic optimization problems in that context. Currently the research goes into many directions. There are many papers devoted to search for the best preconditioner to the problem, see for instance Gratton et al. (2011), Andrea et al. (2017), Fasano and Roma (2016) and many other. Feng et al. (1995) consider the block conjugate gradient methods for solving linear systems with multiple right-hand sides. Block algorithm is also applied to problems with multiple right-hand sides in El Guennouni et al. (2003) (this time the considered matrices are not symmetric). Some researchers concentrate on finding effectively the solution of the unconstrained quadratic optimization or the
associated system of linear equations. For instance in Alléon et al. (2003), Simoncini and Szyld (2007) one may find information on algorithms combining direct methods with the iterative approach. Tasks of the noisy pictures deblurring are formulated frequently in the form of the quadratic programming problems with nonnegativity conditions on variables and regularization, see for instance Chen and Zhou (2010) and references therein. The problem in Chen and Zhou (2010) is essentially quadratic and non smooth due to the regularity term formulated in the $l_1$-norm. Another trend is to find directly factorization of the inverse matrix. Let us mention in that context the papers of Yunlan et al. (1997) and Zhu et al. (2011). In Yunlan et al. (1997) and Zhu et al. (2011) the structure of the hessian matrix is deeply exploited. Let us mention also SelInv – the selected inversion algorithm for the selected inversion of a sparse symmetric matrix developed by Lin et al. (2011). In the first stage of SelInv the $LL^T$ matrix decomposition of $G$ is calculated and in the second stage selected subsets of entries are found. SelInv requires special structure of the matrix to do it effectively. Electronic structure calculations are among possible applications areas. In our result we start from the notion of conjugate vectors and do not use the structure. It is worth to mention here also the iterative refinement technique introduced by Ogita and Oishi (2012).

The content of the paper may be treated as a result on the junction of two disciplines: optimization and linear algebra. Currently a close relationship between optimization and numerical linear algebra algorithms is observed. As O’Leary (2000) stated “The efficiency and effectiveness of most optimization algorithms hinges on the numerical linear algebra algorithms that they utilize. Effective linear algebra is crucial to their success, and because of this, optimization applications have motivated fundamental advances in numerical linear algebra.”

Our paper inscribes into that direction. In our opinion it may be of interest to some researchers working in optimization algorithms and linear algebra as well. In our paper, we use the notion of conjugacy frequently exploited in optimization. We propose a non conventional way of using given set of conjugate directions. In that approach $n$ independent simultaneous directional minimizations starting from the same point $x^0$ are simultaneously executed and finally the solution is obtained by adding the linear combination of the search directions with combination coefficients equal to the calculated directional step sizes to $x^0$. Section 2 contains some basic definitions and properties of the conjugate directions and in consecutive Sect. 3 the main theorem justifying correctness of the proposed procedure using $n$ independent, simultaneous directional searches is formulated and proved. In Sect. 4 the algorithm for recursive generation of the conjugate directions is presented. In Sect. 5 it is shown that the generated conjugate directions form the $RD^{-1}R^T$ decomposition of the inverse of the second order derivative matrix (hessian) – $G^{-1}$. Particular values on the main diagonal matrix $R$ and what is strongly related the main diagonal entries of the $D$ matrix depend on the choice of the parameters controlling the length of the generated vectors. Let us stress that the recursive presentation of the Cholesky factorization presented in the book written by Stoer (1976) (see also Stoer and Bulirsch (1993)) was to some extent an inspiration for us (the same representation was used by Wilkinson (1963) to prove the numerical stability of the Cholesky decomposition). Our method works
also recursively, coming out from a subspace of the given dimension to the subspace of dimension larger by one.

The resulting algorithm is two-phase. Its details are discussed in Sect. 6. In the first phase, we generate set of conjugate directions. The second phase consists of performing \( n \) independent directional minimizations. It may be in our opinion useful in the distributed calculations and cloud computing. The second phase may be also executed by means of simply applying three consecutive matrix/vector multiplications. Both approaches are compared numerically on the set of 128 different strictly convex, quadratic programming problems of different sizes (space dimension varies from 14 to 2000). They were created with the aid of 64 symmetric, positive definite matrices taken from the Florida University repository Davis and Hu (2016) and two different versions of the \( \mathbf{b} \) vector (see Sect. 7). All those matrices were taken from the real life applications and they are stored in the sparse matrix format.

The computational effort in the second phase in our approach would be much lower than in the standard way of decomposing the matrix \( \mathbf{G} \) followed by the solution of two sets of linear equations with triangular matrices and plus eventually a set with the diagonal matrix. The computational cost of the first phase is identical with the cost of calculating the standard \( \mathbf{LDL}^T \) factorization of matrix \( \mathbf{G} \). In the final Sect. 8 some conclusions and corollaries are presented.

2 Some definitions and properties of the conjugate directions

To realize factorization of the inverse \( \mathbf{G}^{-1} \) of the considered symmetric, positive definite matrix \( \mathbf{G} \) we shall construct directions which are conjugate with respect to \( \mathbf{G} \). Formal definition of the \( \mathbf{G} \)-conjugacy is recalled below. Definition of conjugate vectors and proof of their linear independence are taken from Bertsekas (1995).

**Definition 1** Nonzero vectors \( \mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_k \) are conjugate with respect to symmetric, positive definite matrix \( \mathbf{G} \) if and only if

\[
\mathbf{d}_i^T \mathbf{G} \mathbf{d}_j = 0, \quad \forall \ i \neq j
\]

Let us recall that \( \mathbf{G} \)-conjugate directions are linearly independent.

**Lemma 1** If \( \mathbf{G} \) is a symmetric, positive definite, \( n \times n \) matrix, vectors \( \mathbf{d}_i \) (for \( i = 1, \ldots, k \leq n \) are nonzero and mutually conjugate with respect to \( \mathbf{G} \), then they are linearly independent.

**Proof** Let us assume that the thesis is not valid. Then, there exists a convex linear combination of vectors \( \mathbf{d}_i \) equal to \( \mathbf{0} \)

\[
\sum_{i=1}^{k} \alpha^i \mathbf{d}_i = 0 \quad \text{and} \quad \exists \ j, \quad \text{such that} \quad \alpha^j \neq 0.
\]

Let us multiply the equation from both sides by \( \mathbf{d}_j^T \mathbf{G} \). Then,
\[ \sum_{i=1}^{k} \alpha^i d_j Gd_i = \alpha^j d_j Gd_j = 0 \]

due to the \( G \)–conjugacy property. From the last equality we deduce that \( \alpha^j = 0 \), because \( d_j Gd_j > 0 \) by positive definiteness assumption. We obtained the contradiction. That concludes the proof.

3 Solving QP problem by means of simultaneous directional minimizations

The key property used in this section is formulated as the theorem.

**Theorem 1** Let vectors \( d^i \) (for \( i = 1, \ldots, n \)) be a set of directions mutually conjugate with respect to the symmetric, positive definite matrix \( G \), where \( G \) is the second order derivative matrix of problem (1). Let us take any point \( x^0 \in \mathbb{R}^n \) and execute \( n \) exact directional minimization searches along any vector \( d^i \), i.e., solve the following one-dimensional minimization problems

\[
\min_{\alpha \geq 0} \left\{ \bar{f}_i(\alpha) = f(x^0 + \alpha d^i) \right\}, \quad \forall \ i = 1, \ldots, n
\]

and their solutions by \( \alpha^i \) respectively.

Then point \( \bar{x} \) defined as

\[
\bar{x} = x^0 + \sum_{i=1}^{n} \alpha^i d^i
\]

is the optimal solution of the quadratic programming problem (1).

**Proof** We shall show that \( G\bar{x} - b = 0 \). To prove the thesis it is sufficient to show that the gradient \( \bar{g} = \nabla f(\bar{x}) \) (first derivative of function \( f \) calculated at the point \( \bar{x} \)) is orthogonal to all vectors \( d^i \), i.e.,

\[
\bar{g}^T d^i = 0, \quad \forall \ i = 1, \ldots, n.
\]

Let us select any of the conjugate vectors \( d^i \) (let it be \( d^k \)). Then

\[
\left( d^k \right)^T \nabla f(\bar{x}) = \left( d^k \right)^T \left[ G \left( x^0 + \sum_{i=1}^{n} \alpha^i d^i \right) - b \right]
= \left( d^k \right)^T Gx^0 + \sum_{i=1}^{n} \alpha^i \left( d^k \right)^T Gd^i - \left( d^k \right)^T b.
\]
On a conjugate directions method for solving strictly... 529

The conjugacy property ensures that inside the sum there is only one term with nonzero value, namely for $i = k$. Therefore after simple recalculations we obtain

$$
\left( d^k \right)^T \nabla f(\bar{x}) = \left( d^k \right)^T Gx^0 + \alpha^k \left( d^k \right)^T Gd^k - \left( d^k \right)^T b
$$

$$
= \left( d^k \right)^T \left[ G \left( x^0 + \alpha^k d^k \right) - b \right].
$$

Let us notice that

$$
\tilde{f}_i'(\alpha) = \left( \nabla f(x_0 + \alpha d^i) \right)^T d^i
$$

by the rule of differentiating composite functions. Hence

$$
\left( d^k \right)^T \nabla f(\bar{x}) = f^k(\alpha^k) = 0
$$

due to the exact directional minimization on the straight line defined by the set of points $\{x(\alpha) \in \mathbb{R}^n \mid x(\alpha) = x_0 + \alpha d^k, \alpha \in \mathbb{R}^1\}$.

From Lemma 1 we deduce that vectors $d^i$ ($i = 1, \ldots, n$) form the basis of $\mathbb{R}^n$. We proved that the derivative $\nabla f(\bar{x})$ is orthogonal to all vectors from that basis. Therefore, it should be equal to $0$.

Similar theorem may be found in Fletcher (1987). However, he considered it in the essentially sequential way – directional search into one direction, followed by the move to the new point, then another directional minimization from the second point followed by the move to the third point and so on. Fletcher’s theorem is the basis for the conjugate directions methods in optimization. Here, we consider $n$ independent directional minimizations starting all of them from the same point $x_0$.

Theorem 1 suggests an obvious numerical procedure to solve a quadratic programming problem (1) with a symmetric, positive definite matrix $G$ in a computational cloud or on a machine with many parallel processors. We may simply run simultaneously $n$ independent directional minimizations starting from the same point $x^0$. The crucial problem is the effective generation of the conjugate directions. A recursive procedure of their generation is proposed in the consecutive section.

4 Recursive generation of conjugate directions

We shall start with the vector having only one nonzero entry, namely the first one (for instance equal to 1) and every consecutive vector will have one nonzero entry more than the previous one, i.e., $k$-th vector will have first $k$ entries possibly not equal to zero, the rest will assume zero value. Furthermore every $k + 1$ vector should be conjugate to all previous vectors with respect to matrix $G$.

So, let us assume now that the starting vector $d^1 = [c, 0, \ldots, 0]^T$, where $c > 0$ is a positive constant, and try to find consecutive vector $d^2$ verifying the following requirements:
\( d^2 \) is conjugate to \( d^1 \) with respect to \( G \),
- only two first entries of \( d^2 \) may have nonzero values, all others are equal to 0, i.e.,
\( d^2 = [d_1, d_2, 0, \ldots, 0]^T \).

Then
\[
\left( d^1 \right)^T G d^1 = c^2 G_{1,1} = D_1
\]

Defining vector \( d^2 \) as a vector with two nonzero entries as follows
\[
d^2 = [d^2_1, d^2_2, 0, \ldots, 0]^T,
\]
we shall have the following situation
\[
\begin{bmatrix}
    d^1_1 \\
    0 \\
    \vdots \\
    0
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
    d^2_1 \\
    d^2_2 \\
    0 \\
    \vdots \\
    0
\end{bmatrix}.
\]

Vector \( d^2 \) should be \( G \) conjugate with \( d^1 \), i.e.,
\[
\left( d^1 \right)^T G d^2 = 0.
\]

Let us calculate appropriate vector \( d^2 \). Due to the choice of \( d^1 \) and \( d^2 \) we have
\[
\begin{bmatrix}
    d^1_1 \\
    0 \\
    \vdots \\
    0
\end{bmatrix}
\begin{bmatrix}
    G_{1,1} & G_{1,2} & G_{1,3} & \ldots & G_{1,n} \\
    G_{2,1} & G_{2,2} & G_{2,3} & \ldots & G_{2,n} \\
    G_{3,1} & G_{3,2} & G_{3,3} & \ldots & G_{3,n} \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    G_{n,1} & G_{n,2} & G_{n,3} & \ldots & G_{n,n}
\end{bmatrix}
\begin{bmatrix}
    d^2_1 \\
    d^2_2 \\
    0 \\
    \vdots \\
    0
\end{bmatrix}
= \begin{bmatrix}
    d^1_1 \\
    0
\end{bmatrix}
\begin{bmatrix}
    G_{1,1} & G_{1,2} \\
    G_{2,1} & G_{2,2}
\end{bmatrix}
\begin{bmatrix}
    d^2_1 \\
    d^2_2
\end{bmatrix}.
\]

Now, simple matrix calculations lead to
\[
\begin{bmatrix}
    d^1_1 \\
    G_{1,1} & d^1_1 G_{1,1} & G_{1,2}
\end{bmatrix}
\begin{bmatrix}
    d^2_1 \\
    d^2_2
\end{bmatrix}
= d^1_1 G_{1,1} d^2_1 + d^1_1 G_{1,2} d^2_2.
\]

In order to preserve the conjugacy property this quantity should be equal to 0
\[
d^1_1 G_{1,1} d^2_1 + d^1_1 G_{1,2} d^2_2 = 0.
\]
and therefore coefficients of vector $d^2$ should verify the equality

$$G_{1,1}d_1^2 + G_{1,2}d_2^2 = 0. \tag{8}$$

We’ve got one degree of freedom in Eq. (8). We may take for instance $d_2^2 = 1$ and solve Eq. (8) with respect to $d_2^1$.

In what follows we shall derive similar representation for every $k$. In step number $k > 1$ we permit that only the first $k$ elements of the new direction $d^k$ may assume nonzero values, all others are set to 0. Furthermore, we preserve the conjugacy with respect to $G$ to all previous directions $d^1, \ldots, d^{k-1}$ where in every vector $d^i$ ($i < k$) only the first $i$ coefficients may assume nonzero values.

Let us assume the following block representation of matrix $G$

$$G = \begin{bmatrix} G_k & G_r \\ G_r^T & G_{n-k} \end{bmatrix}, \quad G = \begin{bmatrix} G_{k-1} & g & G_r \\ g^T & a & G_r \\ G_r^T & G_{n-k} \end{bmatrix} \tag{9}$$

and let $d^k$ has the following structure

$$d^k = \begin{bmatrix} \bar{d}^k \\ 0 \end{bmatrix}, \quad \text{where} \quad \bar{d}^k = \begin{bmatrix} d \\ c \end{bmatrix} \tag{10}$$

with $d$ being a column vector of size $k - 1$ and $c$ is a scalar.

Similarly $d^i$ ($i < k$) may be represented as

$$d^i = \begin{bmatrix} \bar{d}^i \\ 0 \end{bmatrix}, \tag{11}$$

where $\bar{d}^i$ belongs to the space $\mathbb{R}^{i-1}$.

The conjugacy requirement due to the splitting formulated in Eqs. (9–11) may be written as

$$\left( d^k \right)^T G d^i = \left[ d^T, c \right] \begin{bmatrix} G_{k-1} & g \\ g^T & a \\ G_r^T & G_{n-k} \end{bmatrix} \begin{bmatrix} \bar{d}^i \\ 0 \end{bmatrix} = 0, \quad \forall \ i < k. \tag{12}$$

Hence, the considerations may be restricted to the $k$ dimensional space and conjugacy with respect to the $k \times k$ principal submatrix $G_k$. Applying (9) again we shall arrive to the following equivalent form of (12)

$$\left( d^k \right)^T G d^i = \left[ d^T, c \right] \begin{bmatrix} G_{k-1} & \bar{d}^i \\ g^T & \bar{d}^i \end{bmatrix} = d^T G_{k-1} \bar{d}^i + cg^T \bar{d}^i = 0, \quad \forall \ i < k. \tag{13}$$

Vector $d$ belongs to the space spanned on vectors $d^1, \ldots, d^{k-1}$. Therefore it may be expressed as their linear combination

$$d = \sum_{j=1}^{k-1} \tau_j \bar{d}^j.$$
Let us put the above expression of $d$ to (13) and calculate coefficients $\tau_j$ which will ensure verification of the conjugacy condition by vector $d^k$

\[
(d^k)^T G d^i = \sum_{j=1}^{k-1} \tau_j \left[ \tilde{d}^j \right]^T G_{k-1} \tilde{d}^i + c g^T \tilde{d}^i
\]  

$\forall \ i < k,$ (14)

where $D_i = \left[ \tilde{d}^i \right]^T G_{k-1} \tilde{d}^i$ for each $i < k$. Now, formula (14) results in the following values of coefficients

\[
\tau_i = -c D_i^{-1} \left( \tilde{d}^i \right)^T g
\]  

and finally we obtain the formula specifying new vector

\[
d = - \sum_{j=1}^{k-1} c D_j^{-1} \left( \tilde{d}^j \right)^T g \tilde{d}^j = \sum_{j=1}^{k-1} c \tilde{d}^j D_j^{-1} \left( \tilde{d}^j \right)^T g.
\]  

(16)

Hence, we may formulate the following algorithm for generation of vectors mutually conjugate with respect to matrix $G$.

**Algorithm I**

[Initialization] Set $k = 1$. Select $c$, for instance $c = 1$. Build vector

\[
d^1 = \left[ \begin{array}{c} c \\ 0 \end{array} \right].
\]

Calculate $D_1 = (d^1)^T G_{11} d^1$.

[Loop]

Increase the dimension, set $k = k+1$.

while ($k < n$)

{

1. Calculate vector $d = - \sum_{j=1}^{k-1} c \tilde{d}^j D_j^{-1} \left( \tilde{d}^j \right)^T g$.
2. Build vector $d^k$ as follows $\left( d^k \right)^T = [d^T, c, 0]$.
3. Calculate $D_k = (d^k)^T G_{kk} d^k$.
4. Set $k = k + 1$.

}

The induction analysis presented above proves the following lemma

**Lemma 2** Vectors $d^1$, $d^2$, ..., $d^n$ generated by Algorithm I are mutually conjugate with respect to matrix $G$. 

\[ Springer \]
4.1 Matrix form of the algorithm for generating conjugate directions

Algorithm presented above may be written in the matrix form.

Let us denote by $\mathbf{R}^i$ ($1 \leq i \leq k - 1$) the rectangular matrix with columns equal to consecutive $\mathbf{d}^j$ ($1 \leq j \leq i$) vectors and $\bar{\mathbf{R}}^i$ ($1 \leq i \leq k - 1$) the rectangular matrix with columns equal to consecutive $\bar{\mathbf{d}}^j$ ($1 \leq j \leq i$), i.e.,

$$\mathbf{R}^i = \begin{bmatrix} \mathbf{d}^1 & \cdots & \mathbf{d}^i \end{bmatrix} \quad \text{and} \quad \bar{\mathbf{R}}^i = \begin{bmatrix} \bar{\mathbf{d}}^1 & \cdots & \bar{\mathbf{d}}^i \end{bmatrix}.$$  

Due to their construction

$$\mathbf{R}^i = \begin{bmatrix} \bar{\mathbf{R}}^i \\ \mathbf{0} \end{bmatrix}$$

and

$$\left( \mathbf{R}^k \right)^T \mathbf{G} \mathbf{R}^k = \left( \bar{\mathbf{R}}^k \right)^T \mathbf{G}_k \bar{\mathbf{R}}^k = \mathbf{D}_k = \text{diag}\{D_1, D_2, \ldots, D_k\}.$$  

Taking the above relation into account we may express formula (16) in the following matrix form

$$\mathbf{d} = -c \bar{\mathbf{R}}^{k-1} \mathbf{D}_k^{-1} \left( \bar{\mathbf{R}}^{k-1} \right)^T \mathbf{g}. \quad (17)$$

Let us introduce the following notation

$$\mathbf{L} = \left( \bar{\mathbf{R}}^{k-1} \right)^T.$$  

Then formula (17) may be expressed as

$$\mathbf{d} = -c \mathbf{L}^T \mathbf{D}_k^{-1} \mathbf{L} \mathbf{g}. \quad (18)$$

Below is the formal formulation of the pseudo code of the matrix form of Algorithm I.

**Algorithm II**

[Initialization] Set $k = 1$. Set $\mathbf{L}^1 = 1$, select $c$, for instance $c = 1$.

[Loop]

while $(k < n)$

1. Calculate $\mathbf{D} = \mathbf{L}^k \mathbf{G}_k \left( \mathbf{L}^k \right)^T$.
2. Increase the dimension by 1; set $k = k + 1$. Calculate vector $\mathbf{d}$ using formula (18).
3. Build new matrix $\mathbf{L}^k$ as follows

$$\mathbf{L}^k = \begin{bmatrix} \mathbf{L}^{k-1} & \mathbf{0} \\ \mathbf{d}^T & c \end{bmatrix}.$$  

}
To calculate vector $\mathbf{d}$ we should realize the matrix/vector multiplication, followed by the eventual multiplication of each component by scalar $c$ and consecutive division by the main-diagonal entry of the diagonal matrix $\mathbf{D}$. The last two operations can be realized simultaneously.

New lower triangular matrix $\mathbf{L}^k$ is augmented at each step by the addition of a new row representing transposed vector $\overline{\mathbf{d}}^k$ and a new column containing exclusively entries equal to 0 above the main diagonal.

5 Generated directions and their relation with the factorization of the inverse matrix

Last matrix $\mathbf{L}^n$ contains $n$-rows being transpositions of $n$ mutually $\mathbf{G}$-conjugate vectors and verifies the following matrix relation

$$\mathbf{L}^n \mathbf{G} \left[ \mathbf{L}^n \right]^T = \mathbf{D}^n, \quad (19)$$

where $\mathbf{L}^n$ is lower triangular and $\mathbf{D}^n$ is the diagonal matrix.

Below we shall show that Algorithm II produces the LDL factorization of the inverse of $\mathbf{G}$.

**Lemma 3** The following equality holds

$$\mathbf{G}^{-1} = \left[ \mathbf{L}^n \right]^T \left[ \mathbf{D}^n \right]^{-1} \mathbf{L}^n = \mathbf{R}^n \left[ \mathbf{D}^n \right]^{-1} \left[ \mathbf{R}^n \right]^T. \quad (20)$$

**Proof** Taking the inverses in Eq. (19) we obtain

$$\left[ \mathbf{L}^n \right]^{-T} \mathbf{G}^{-1} \left[ \mathbf{L}^n \right]^{-1} = \left[ \mathbf{D}^n \right]^{-1}. \quad (21)$$

Multiplying (21) from the left-hand side by $\left[ \mathbf{L}^n \right]^T$ and by $\mathbf{L}^n$ from the right-hand side we shall obtain the desired expression (20) of the inverse of $\mathbf{G}$.

It means that the proposed algorithm produces directly an $\mathbf{RD}^{-1}\mathbf{R}^T$ factorization of the inverse matrix $\mathbf{G}^{-1}$. It is easily observed that the same statement is valid for the intermediate set of generated vectors.

**Lemma 4** The following equality holds for each $1 \leq k \leq n$

$$\mathbf{G}_{kk}^{-1} = \left[ \mathbf{L}^k \right]^T \left[ \mathbf{D}^k \right]^{-1} \mathbf{L}^k = \mathbf{R}^k \left[ \mathbf{D}^k \right]^{-1} \left[ \mathbf{R}^k \right]^T. \quad (22)$$

Deeper analysis and stability issues of this approach to obtain directly the factorization of the inverse matrix will be pursued elsewhere.
6 Two-phase algorithm

A two-phase algorithm is proposed on the basis of the above theoretical considerations. In the first phase we calculate the conjugate directions. For their calculation we may apply either Algorithm I or Algorithm II formulated above. As follows from the above discussion the proposed algorithm is recursive. We start with the space of dimension equal to 1 and increase it by one at each step. It is also important in this procedure that the only algebraic operations involved are the matrix/vector multiplication and division or multiplication of the matrix entries by an appropriate number.

It is possible to reduce further the computational effort, if we restrict calculation of the entries of the diagonal matrix $D$ in step 1 to the last entry on the main diagonal, i.e., $D_{kk}$. All other entries on the main diagonal are the same as in the previous iteration.

In the second phase the optimal solution of the QP (quadratic programming) problem (1) is found. After closing calculation of the conjugate gradients it is clear, whether the QP problem has the optimal solution or it is unbounded. So, let us now assume that the QP problem is bounded from below.

The algorithm executing $n$ independent simultaneous directional minimizations is presented in Sect. 6.1. It is based on theorem 1.

Second possibility presented in Sect. 6.2 is to solve the stationarity condition. By Lemma 4 in Sect. 5 we know that as a byproduct of the conjugate gradient generation we obtain the factorization of the inverse hessian. Therefore we can find the solution of the stationarity condition (3) by simple matrix/vector multiplications. This approach is discussed in the Sect. 6.2.

6.1 Finding the solution of the QP problem

The key property used in this subsection is the result of Theorem 1.

Similar theorem may be found in Fletcher (1987). However, he considered it in the essentially sequential way – directional search into one direction, followed by the move to the new second point, then another directional minimization from the second point followed by the move to the third point and so on. Fletcher’s theorem is the basis for the conjugate directions methods in nonlinear unconstrained optimization. Here, we consider $n$ independent directional minimizations starting all of them from the same point $x^0$.

Theorem 1 suggests an obvious numerical procedure to find the minimum of the quadratic function (1) with a symmetric, positive definite matrix $G$ in a computational cloud or on a machine with many parallel processors. We may simply run simultaneously $n$ independent directional minimizations starting from the same point $x^0$.

Algorithm III

A0. Select the starting point $x^0$. $x^0 = 0$ may be a suitable choice, because then $\nabla f(x^0) = -b$.

A1. Execute $n$ independent directional minimizations, i.e., find solutions to all problems
\[
\min_{\alpha} \left\{ \bar{f}^i(\alpha) = f(x^0 + \alpha d^i) \right\}, \quad \forall \ i = 1, \ldots, n. \tag{23}
\]

A2. Calculate the solution of the problem using the formula

\[
\bar{x} = x^0 + \sum_{i=1}^{n} \alpha^i d^i. \tag{24}
\]

Function \( f \) in formula (23) is quadratic (see formula (1)). Therefore, each directional step size \( \alpha^i \) is the stationary point of the derivative of function \( \bar{f}^i \), which is minimized in Step A1. Hence the step size verifies the following equation

\[
d\bar{f}^i(d\alpha (\alpha^i)) = (G(x^0 + \alpha^i d^i) - b)^T d^i = 0. \tag{25}
\]

Simple arithmetic operations on Eq. (25) leads to the following expression of \( \alpha^i \)

\[
\alpha^i = -\frac{(Gx^0 - b)^T d^i}{(d^i)^T Gd^i} = -\frac{((\nabla f(x^0))^T d^i}{(d^i)^T Gd^i}. \tag{26}
\]

What is important, derivative \( \nabla f(x^0) \) may be calculated once. The same value is used for every step \( i \). We need the following data: \( \nabla f(x^0), d^i \) and \( G \) to calculate \( \alpha^i \).

6.2 Finding the solution of the set of equations

Conjugacy condition may be equivalently written as follows

\[
R^T GR = D. \tag{27}
\]

Multiplying Eq. (27) from the left-hand side by the inverse of \( R^T \) and from the right-hand side by \( R^{-1} \) we shall obtain

\[
G = R^{-T} D R^{-1}. \tag{28}
\]

Now application of the commonly known rule of calculating the inverse of the product of square nonsingular matrices yields

\[
G^{-1} = R D^{-1} R^T. \tag{29}
\]

Let us apply formula (29) to equation

\[
Gx = b.
\]

This results in

\[
x = G^{-1} b = RD^{-1} R^T b. \tag{30}
\]
On a conjugate directions method for solving strictly... 537

Table 1  Problems of size from 14 to 500

| No. | Id  | Group    | Name    | Size | Non zeros | Type                     |
|-----|-----|----------|---------|------|-----------|--------------------------|
| 1   | 1440| Oberwolfach | LFAT5  | 14   | 46        | Model reduction          |
| 2   | 1438| Oberwolfach | LF10   | 18   | 82        | Model reduction          |
| 3   | 436 | FIDAP    | ex5    | 27   | 279       | Computational fluid dynamics |
| 4   | 23  | HB       | bcsstk01 | 48  | 400       | Structural               |
| 5   | 872 | Pothen   | mesh1e1 | 48   | 306       | Structural               |
| 6   | 873 | Pothen   | mesh1em1 | 48   | 306       | Structural               |
| 7   | 874 | Pothen   | mesh1em6 | 48   | 306       | Structural               |
| 8   | 24  | HB       | bcsstk02 | 66   | 4356      | Structural               |
| 9   | 57  | HB       | bcsstm02 | 66   | 66        | Structural               |
| 10  | 220 | HB       | nos4    | 100  | 594       | Structural               |
| 11  | 25  | HB       | bcsstk03 | 112  | 640       | Structural               |
| 12  | 26  | HB       | bcsstk04 | 132  | 3648      | Structural               |
| 13  | 44  | HB       | bcsstk22 | 138  | 696       | Structural               |
| 14  | 72  | HB       | bcsstm22 | 138  | 138       | Structural               |
| 15  | 206 | HB       | lund_a  | 147  | 2449      | Structural               |
| 16  | 207 | HB       | lund_b  | 147  | 2441      | Structural               |
| 17  | 27  | HB       | bcsstk05 | 153  | 2423      | Structural               |
| 18  | 60  | HB       | bcsstm05 | 153  | 153       | Structural               |
| 19  | 217 | HB       | nos1    | 237  | 1017      | Structural               |
| 20  | 877 | Pothen   | mesh3e1 | 289  | 1377      | Structural               |
| 21  | 878 | Pothen   | mesh3em5 | 289  | 1377      | Structural               |
| 22  | 875 | Pothen   | mesh2e1 | 306  | 2018      | Structural               |
| 23  | 876 | Pothen   | mesh2em5 | 306  | 2018      | Structural               |
| 24  | 229 | HB       | plat362 | 362  | 5786      | 2D/3D                    |
| 25  | 315 | Bai      | mhdb416 | 416  | 2312      | Electromagnetics         |
| 26  | 28  | HB       | bcsstk06 | 420  | 7860      | Structural               |
| 27  | 29  | HB       | bcsstk07 | 420  | 7860      | Duplicate structural     |
| 28  | 61  | HB       | bcsstm06 | 420  | 420       | Structural               |
| 29  | 62  | HB       | bcsstm07 | 420  | 7252      | Structural               |
| 30  | 221 | HB       | nos5    | 468  | 5172      | Structural               |
| 31  | 42  | HB       | bcsstk20 | 485  | 3135      | Structural               |
| 32  | 70  | HB       | bcsstm20 | 485  | 485       | Structural               |
| 33  | 2   | HB       | 494_bus | 494  | 1666      | Power network            |

In view of equality (30) it is easy to distribute the calculation of \( \mathbf{x} \) verifying Eq. (1). It involves exclusively matrix/vector multiplications and calculation of the image of the matrix \( \mathbf{D}^{-1} \) on some appropriate vector. Matrix \( \mathbf{D} \) is diagonal. Hence multiplication by \( \mathbf{D}^{-1} \) requires only \( n \) arithmetic divisions. There are efficient methods to realize matrix/vector multiplication effectively, see for instance Bertsekas and Tsitsiklis (1989).
Table 2 Problems of size from 500 to 2000

| No. | Id  | Group | Name           | Size | Non nonzeros | Type            |
|-----|-----|-------|----------------|------|--------------|-----------------|
| 34  | 339 | Boeing| bcsstk34       | 588  | 21418        | Structural      |
| 35  | 3   | HB    | 662_bus        | 662  | 2474         | Power network   |
| 36  | 222 | HB    | nos6           | 675  | 3255         | 2D/3D          |
| 37  | 4   | HB    | 685_bus        | 685  | 3249         | Power network   |
| 38  | 357 | Boeing| msc00726       | 726  | 34518        | Structural      |
| 39  | 223 | HB    | nos7           | 729  | 4617         | 2D/3D          |
| 40  | 41  | HB    | bcsstk19       | 817  | 6853         | Structural      |
| 41  | 69  | HB    | bcsstm19       | 817  | 817          | Structural      |
| 42  | 218 | HB    | nos2           | 957  | 4137         | Structural      |
| 43  | 219 | HB    | nos3           | 960  | 15844        | Structural      |
| 44  | 358 | Boeing| msc01050       | 1050 | 26198        | Structural      |
| 45  | 30  | HB    | bcsstk08       | 1074 | 12960        | Structural      |
| 46  | 63  | HB    | bcsstm08       | 1074 | 1074         | Structural      |
| 47  | 31  | HB    | bcsstk09       | 1083 | 18437        | Structural      |
| 48  | 64  | HB    | bcsstm09       | 1083 | 1083         | Structural      |
| 49  | 32  | HB    | bcsstk10       | 1086 | 22070        | Structural      |
| 50  | 1   | HB    | 1138_bus       | 1138 | 4054         | Power network   |
| 51  | 49  | HB    | bcsstk27       | 1224 | 56126        | Structural      |
| 52  | 1911 | TKK  | plbuckle       | 1282 | 30644        | Structural      |
| 53  | 359 | Boeing| msc01440       | 1440 | 44998        | Structural      |
| 54  | 33  | HB    | bcsstk11       | 1473 | 34241        | Structural      |
| 55  | 34  | HB    | bcsstk12       | 1473 | 34241        | Duplicate structural |
| 56  | 66  | HB    | bcsstm11       | 1473 | 1473         | Structural      |
| 57  | 67  | HB    | bcsstm12       | 1473 | 19659        | Structural      |
| 58  | 430 | FIDAP | ex33           | 1733 | 22189        | Computational fluid dynamics |
| 59  | 36  | HB    | bcsstk14       | 1806 | 63454        | Structural      |
| 60  | 427 | FIDAP | ex3            | 1821 | 52685        | Computational fluid dynamics |
| 61  | 757 | NASA  | nasa1824       | 1824 | 39208        | Structural      |
| 62  | 228 | HB    | plat1919       | 1919 | 32399        | 2D/3D          |
| 63  | 48  | HB    | bcsstk26       | 1922 | 30336        | Structural      |
| 64  | 76  | HB    | bcsstm26       | 1922 | 1922         | Structural      |

7 Numerical experiments

The sample matrix test problems from the repository of the University of Florida were used for the numerical verification of the proposed algorithm (description may be found in Davis and Hu (2011)). The repository is available on-line and maintained by Davis and Hu (2016). We restricted calculations to the subset of symmetric, positive definite matrices from the repository. They are stored in three different formats: Rutherford-Boeing format (see Duff et al. (1989a, b, 1997)), in the Matrix Market format Boisvert
et al. (1997a) and Boisvert et al. (1997b) and the MATLAB format. In the MATLAB format used by us, the problem set is a single MATLAB variable of struct type, stored in a single MAT-file.

Tables 1 and 2 contain information about the test problems. We selected from the repository all problems with symmetric, positive definite matrices of size between 0 and 2000. Many of them are structural problems, where matrix $G$ is the so-called
Table 4  Results for problems with size from 501 to 2000 (all components of $b$ are equal to 1)

| No. | Size  | Fct-time | AccFct | $FM_t$ | AccSol | $FD_t$ | AccSol |
|-----|-------|----------|--------|--------|--------|--------|--------|
| 34  | 588   | 2.765625 | 1.18e-08 | 0 | 1.47e-13 | 0.421875 | 1.47e-13 |
| 35  | 662   | 2        | 3.88e-13 | 0 | 7.28e-11 | 0.265625 | 7.28e-11 |
| 36  | 675   | 2.59375  | 5.84e-09 | 0 | 7.58e-09 | 0.328125 | 7.58e-09 |
| 37  | 685   | 2.5      | 2.47e-12 | 0 | 5.82e-11 | 0.390625 | 5.82e-11 |
| 38  | 726   | 5.296875 | 6.33e-07 | 0 | 1.17e-12 | 0.75    | 1.17e-12 |
| 39  | 729   | 4.96875  | 3.26e-09 | 0 | 9.54e-07 | 0.765625 | 9.54e-07 |
| 40  | 817   | 4.6875   | 1.81e+03 | 0 | 1.21e-04 | 0.796875 | 1.21e-04 |
| 41  | 817   | 2.109375 | 0        | 0 | 1.11e-16 | 0.46875 | 1.11e-16 |
| 42  | 957   | 7.078125 | 1.79e-04 | 0.03125 | 1.33e-06 | 0.953125 | 1.33e-06 |
| 43  | 960   | 11.3125  | 2.01e-13 | 0 | 1.64e-11 | 1.703125 | 1.64e-11 |
| 44  | 1050  | 3.890625 | 2.97e-08 | 0 | 3.48e-01 | 0.328125 | 3.48e-01 |
| 45  | 1074  | 11.6125  | 5.28e-05 | 0.03125 | 7.28e-11 | 2.328125 | 7.28e-11 |
| 46  | 1074  | 3.640625 | 0        | 0 | 1.11e-16 | 0.625   | 1.11e-16 |
| 47  | 1083  | 16.296875| 3.51e-08 | 0 | 1.32e-11 | 2.40625 | 1.32e-11 |
| 48  | 1083  | 3.71875  | 0        | 0 | 0        | 0.03125 | 0 |
| 49  | 1086  | 9.796875 | 4.53e-09 | 0 | 2.78e-11 | 1.296875 | 2.78e-11 |
| 50  | 1138  | 9.625    | 1.64e-11 | 0.03125 | 6.52e-09 | 1.4375  | 6.52e-09 |
| 51  | 1224  | 23.75    | 8.7e-10  | 0.03125 | 7.72e-13 | 3.4375  | 7.72e-13 |
| 52  | 1282  | 15.359375| 2.31e-10 | 0 | 5.37e-12 | 2.046875 | 5.37e-12 |
| 53  | 1440  | 9.65625  | 4.58e-09 | 0 | 1.54e-12 | 1.734375 | 1.54e-12 |
| 54  | 1473  | 30.515625| 3.3e+07  | 0.03125 | 5.75e+02 | 4.421875 | 5.75e+02 |
| 55  | 1473  | 30.453125| 3.3e+07  | 0 | 5.75e+02 | 4.421875 | 5.75e+02 |
| 56  | 1473  | 6.796875 | 0        | 0 | 1.11e-16 | 0.078125 | 1.11e-16 |
| 57  | 1473  | 10.328125| 7.91e-14 | 0 | 2.82e-10 | 0.796875 | 2.82e-10 |
| 58  | 1733  | 32.328125| 5.99e+10 | 0 | 3.15e+04 | 4.71875 | 3.15e+04 |
| 59  | 1806  | 72.609375| 5.71e-06 | 0.03125 | 2.73e-11 | 11.046875| 2.73e-11 |
| 60  | 1821  | 77.03125 | 4.61e+07 | 0.0625 | 2.08e+05 | 12.125  | 2.08e+05 |
| 61  | 1824  | 77.890625| 2.49e-08 | 0.0625 | 1.89e-10 | 12.171875| 1.89e-10 |
| 62  | 1919  | 95.9065  | 1.11e+01 | 0.0625 | 8.98e+04 | 14.203125| 6.98e+04 |
| 63  | 1922  | 66.640625| 4.91e-04 | 0.03125 | 6.39e-09 | 10.953125| 6.39e-09 |
| 64  | 1922  | 11.546875| 0        | 0 | 1.11e-16 | 0.09375 | 1.11e-16 |

stiffness matrix of the structure and is usually symmetric and positive definite. It arises as the result of the finite element discretization of the problem. The details may be found for instance in Felippa (2001) or McGuire et al. (2000).

The computational results are presented in Tables 3–6. The quantities defined below do appear in particular columns:
Table 5  Results for problems with size from 14 to 500 (all components of \( x \) are equal to 1)

| No. | Size | \( FM_t \) | AccSol | \( FD_t \) | AccSol | Xsol |
|-----|------|------------|--------|------------|--------|------|
| 1   | 14   | 0          | 3.87e−12 | 0          | 3.87e−12 | 1.22e−13 |
| 2   | 18   | 0          | 4.37e−11 | 0          | 4.37e−11 | 9.55e−13 |
| 3   | 27   | 0          | 1.16e−09 | 0          | 1.16e−09 | 3.66e−10 |
| 4   | 48   | 0          | 3.34e−06 | 0          | 3.34e−06 | 2.26e−12 |
| 5   | 48   | 0          | 3.55e−15 | 0.015625   | 3.55e−15 | 6.66e−16 |
| 6   | 48   | 0          | 7.11−15  | 0.015625   | 7.11e−15 | 1.55e−15 |
| 7   | 48   | 0          | 3.55e−15 | 0          | 3.55e−15 | 7.77e−15 |
| 8   | 66   | 0          | 8.19e−12 | 0.03125    | 8.19e−15 | 2.83e−14 |
| 9   | 66   | 0          | 0        | 0          | 0       | 0    |
| 10  | 100  | 0          | 2.74e−16 | 0.015625   | 2.74e−16 | 5.55e−15 |
| 11  | 112  | 0          | 2.75e−04 | 0.03125    | 2.75e−04 | 4.46e−12 |
| 12  | 132  | 0          | 9.31e−09 | 0.03125    | 9.31e−09 | 1.9e−12 |
| 13  | 138  | 0          | 3.73e−09 | 0          | 3.73e−09 | 5.84e−13 |
| 14  | 138  | 0          | 0        | 0          | 0       | 0    |
| 15  | 147  | 0          | 3.87e−07 | 0.015625   | 3.87e−07 | 6.13e−12 |
| 16  | 147  | 0          | 3.64e−12 | 0.03125    | 3.64e−12 | 3.72e−14 |
| 17  | 153  | 0          | 3.06e−09 | 0.03125    | 3.06e−09 | 3.49e−14 |
| 18  | 153  | 0          | 0        | 0          | 0       | 0    |
| 19  | 237  | 0          | 2.18e−06 | 0.015625   | 2.18e−06 | 6.48e−12 |
| 20  | 289  | 0          | 1.07e−14 | 0.078125   | 1.07e−14 | 2.55e−15 |
| 21  | 289  | 0          | 1.07e−14 | 0.015625   | 1.07e−14 | 3.22e−15 |
| 22  | 306  | 0          | 3.98e−13 | 0.0625     | 3.98e−13 | 8.55e−14 |
| 23  | 306  | 0          | 2.27e−13 | 0.0625     | 2.27e−13 | 9.05e−14 |
| 24  | 362  | 0          | 2.82e−01 | 0.125      | 2.82e−01 | 6.32e+03 |
| 25  | 416  | 0          | 1.33e−15 | 0.03125    | 1.33e−15 | 5.05e−09 |
| 26  | 420  | 0          | 5.94e−05 | 0.140625   | 5.94e−05 | 6.00e−11 |
| 27  | 420  | 0          | 5.94e−05 | 0.15625    | 5.94e−05 | 6.00e−11 |
| 28  | 420  | 0          | 0        | 0.03125    | 0        | 0    |
| 29  | 420  | 0          | 1.93e−12 | 0.171875   | 1.93e−12 | 1.75e−14 |
| 30  | 468  | 0.3125     | 9.31e−10 | 0.203125   | 9.31e−10 | 4.00e−14 |
| 31  | 485  | 0          | 1.11e+03 | 0.21875    | 1.11e+03 | 2.44e−06 |
| 32  | 485  | 0          | 0        | 0.03125    | 0        | 0    |
| 33  | 494  | 0          | 7.28e−12 | 0.109375   | 7.28e−12 | 8.55e−13 |

Fct-time  time of factorization in seconds of CPU,
AccFct   accuracy of factorization max\(_{ij} (|L * G * L^T - D|)_{ij}|,
\( FM_t \)  time of finding \( x \) via the matrix/vector operations,
AccSolM accuracy of the solution (max\(_i |(G * x - b)_i|\)),
obtained with the aid of matrix/vector operations
\( FD_t \)  time of finding \( x \) via \( n \) consecutive directional minimizations,
Table 6  Results for problems with size from 501 to 2000 (all components of x are equal to 1)

| No. | Size | $FM_t$ | AccSol | $FD_t$ | AccSol | Xsol |
|-----|------|--------|--------|--------|--------|------|
| 34  | 588  | 0      | 2.51e−08 | 0.421875 | 2.51e−08 | 4.34e−14 |
| 35  | 662  | 0      | 1.59e−12 | 0.265635 | 1.59e−12 | 2.88e−14 |
| 36  | 675  | 0      | 9.08e−09 | 0.34375 | 9.08e−09 | 1.32e−09 |
| 37  | 685  | 0      | 2.18e−11 | 0.390625 | 2.18e−11 | 4.13e−14 |
| 38  | 726  | 0      | 2.92e−06 | 0.765625 | 2.92e−06 | 3.35e−14 |
| 39  | 729  | 0.03125 | 5.12e−09 | 0.734375 | 5.12e−09 | 6.70e−11 |
| 40  | 817  | 0      | 1.81e+03 | 0.78125 | 1.67e+03 | 8.85e−07 |
| 41  | 817  | 0      | 0       | 0.046875 | 0       | 0    |
| 42  | 957  | 0.03125 | 4.11e−04 | 0.9375 | 4.11e−04 | 5.93e−10 |
| 43  | 960  | 0.3125 | 6.54e−13 | 1.6875 | 6.54e−13 | 3.27e−10 |
| 44  | 1050 | 0      | 2.89e−08 | 0.328125 | 2.89e−08 | 1.53e−03 |
| 45  | 1074 | 0.03125 | 1.22e−04 | 2.296875 | 1.22e−04 | 1.44e−11 |
| 46  | 1074 | 0      | 0       | 0.015625 | 0       | 0    |
| 47  | 1083 | 0      | 3.09e−07 | 2.40625 | 3.09e−07 | 6.06e−13 |
| 48  | 1083 | 0      | 0       | 0.046875 | 0       | 0    |
| 49  | 1086 | 0      | 3.73e−08 | 1.296875 | 3.73e−08 | 3.64e−13 |
| 50  | 1138 | 0      | 2.36e−11 | 1.46875 | 2.36e−11 | 6.62e−13 |
| 51  | 1224 | 0.03125 | 5.33e−09 | 3.453125 | 5.33e−09 | 6.46e−14 |
| 52  | 1282 | 0.03125 | 2.63e−09 | 2       | 2.63e−09 | 1.17e−13 |
| 53  | 1440 | 0      | 4.82e−09 | 1.703125 | 4.82e−09 | 2.01e−11 |
| 54  | 1473 | 0      | 7.99e+07 | 4.453125 | 7.99e+07 | 6.71e+02 |
| 55  | 1473 | 0.03125 | 7.99e+07 | 4.4375 | 7.99e+07 | 6.71e+02 |
| 56  | 1473 | 0      | 0       | 0.0625 | 0       | 0    |
| 57  | 1473 | 0      | 5.2e−14 | 0.8125 | 5.2e−14 | 1.52e−10 |
| 58  | 1733 | 0.015625 | 4.41e+10 | 4.71875 | 4.41e+10 | 6.23e+04 |
| 59  | 1806 | 0.03125 | 3.81e−05 | 11.03125 | 3.81e−05 | 2.01e−12 |
| 60  | 1821 | 0.0625 | 4.92e+07 | 12.109375 | 4.92e+07 | 2.87e+04 |
| 61  | 1824 | 0.0625 | 3.27e−08 | 12.171875 | 3.27e−08 | 3.66e−12 |
| 62  | 1919 | 0.03125 | 9.94e+00 | 14.203125 | 9.94e+00 | 4.65e+04 |
| 63  | 1922 | 0.03125 | 5.44e−04 | 10.953125 | 5.44e−04 | 5.11e−10 |
| 64  | 1922 | 0      | 0       | 0.09375 | 0       | 0    |

AccSolD accuracy of the solution $\max_i |(G * x - b)_i|$, obtained with the aid of $n$ consecutive directional minimizations.

Tables 3 and 4 contain the results for problems with the right-hand side vector $b = [1, 1, \ldots, 1]^T$. Unfortunately, almost all problems in the repository do not contain vector b.

Tables 5 and 6 contain results for problems with the same matrices $G$ as above. The only difference is that the solution vector of the linear system is assumed to have
Table 7  Results for problems with size from 14 to 500 (application of classical Cholesky factorization)

| No. | Size | Fct-time | AccFct | \( F_{1t} \) | AccSol1 | \( F_{2t} \) | AccSol2 |
|-----|------|----------|--------|-------------|--------|-------------|--------|
| 1   | 14   | 0        | 9.09e−13 | 0           | 1.86e−09 | 0           | 1.95e−14 |
| 2   | 18   | 0        | 2.91e−11 | 0.015625    | 3.37e−11 | 0           | 2.84e−13 |
| 3   | 27   | 0        | 4.66e−10 | 0           | 7.57e−10 | 0           | 1.02e−08 |
| 4   | 48   | 0        | 4.77e−07 | 0           | 9.54e−07 | 0.03125    | 3.41e−13 |
| 5   | 48   | 0        | 3.55e−15 | 0           | 5.33e−15 | 0           | 4.44e−16 |
| 6   | 48   | 0        | 5.33−15  | 0.015625    | 7.99e−15 | 0           | 1.33e−15 |
| 7   | 48   | 0        | 2.66e−15 | 0           | 2.66e−15 | 0           | 4.44e−16 |
| 8   | 66   | 0        | 1.82e−12 | 0           | 4.70e−12 | 0.03125    | 3.41e−13 |
| 9   | 66   | 0        | 2.78e−17 | 0           | 2.78e−17 | 0           | 3.33e−16 |
| 10  | 100  | 0        | 2.22e−16 | 0           | 1.80e−16 | 0.03125    | 2.27e−13 |
| 11  | 112  | 0        | 1.53e−05 | 0           | 6.10e−05 | 0           | 1.07e−11 |
| 12  | 132  | 0        | 2.79e−09 | 0           | 5.59e−09 | 0.015625   | 1.53e−12 |
| 13  | 138  | 0        | 9.31e−10 | 0.03125     | 1.86e−09 | 0           | 2.30e−12 |
| 14  | 138  | 0        | 8.67e−19 | 0.015625    | 1.73e−18 | 0           | 2.22e−16 |
| 15  | 147  | 0        | 3.19e−07 | 0.03125     | 1.49e−07 | 0.015625   | 3.34e−11 |
| 16  | 147  | 0        | 1.36e−12 | 0           | 1.36e−12 | 0.03125    | 8.90e−14 |
| 17  | 153  | 0        | 4.66e−10 | 0.015625    | 1.66e−09 | 0           | 1.25e−12 |
| 18  | 153  | 0        | 1.11e−16 | 0           | 2.22e−16 | 0           | 2.22e−16 |
| 19  | 237  | 0        | 2.38e−07 | 0.03125     | 4.02e−07 | 0.015625   | 6.91e−10 |
| 20  | 289  | 0.015625 | 2.66e−15 | 0.015625    | 3.55e−15 | 0.046875   | 6.66e−16 |
| 21  | 289  | 0        | 8.88e−16 | 0.0625      | 1.78e−15 | 0.03125    | 3.33e−16 |
| 22  | 306  | 0.015625 | 2.84e−13 | 0.015625    | 2.84e−13 | 0.03125    | 5.33e−15 |
| 23  | 306  | 0        | 4.26e−14 | 0.046875    | 1.14e−13 | 0.03125    | 2.44e−15 |
| 24  | 362  | 0.015625 | 2.50e−16 | 0.046875    | 4.96e−16 | 0.09375    | 8.61e−07 |
| 25  | 416  | 0        | 2.22e−16 | 0.046875    | 4.44e−16 | 0.03125    | 2.56e−13 |
| 26  | 420  | 0        | 1.91e−06 | 0.046875    | 3.34e−06 | 0.03125    | 1.62e−11 |
| 27  | 420  | 0        | 1.91e−06 | 0.046875    | 3.34e−06 | 0.046875   | 1.62e−11 |
| 28  | 420  | 0        | 9.09e−13 | 0.015625    | 1.82e−12 | 0.0625     | 2.22e−16 |
| 29  | 420  | 0        | 1.59e−12 | 0.046875    | 1.82e−12 | 0.046875   | 3.55e−14 |
| 30  | 468  | 0.3125   | 1.16e−10 | 0.09375     | 2.91e−10 | 0.09375    | 9.34e−13 |
| 31  | 485  | 0        | 2        | 0.03125     | 2        | 0.015625   | 4.67e−06 |
| 32  | 485  | 0        | 7.45e−09 | 0.03125     | 7.45e−09 | 0.046875   | 3.33e−16 |
| 33  | 494  | 0        | 1.27e−11 | 0.046875    | 1.64e−11 | 0.046875   | 1.05e−09 |

all components equal to one \( (\hat{x}_i = 1, \quad \forall i) \) and vector \( \mathbf{b} = \mathbf{G}\hat{x} \) respectively. We omitted the factorization accuracy and the decomposition times. They are the same as in Tables 3 and 4 presented above.

Below, in Tables 7 and 8 we present the results obtained by means of the standard Cholesky decomposition of matrix \( \mathbf{G} = \mathbf{L}\mathbf{L}^T \). Having this factorization available two sets of linear equations with triangular matrices are solved. The first one by means
of the forward substitution, the second by means of the backward substitution. Those calculations were suggested by one of the reviewers for comparison purposes. Some quantities stored in columns of Tables 7 and 8 have different meaning than in previous tables. Therefore, they are redefined below.

| No. | Size | Fct-time | AccFct  | $F1_t$ | AccSol1 | $F2_t$ | AccSol2 |
|-----|------|----------|---------|--------|---------|--------|---------|
| 34  | 588  | 0        | 5.59e−09 | 0.078125 | 1.02e−08 | 0.140625 | 2.66e−14 |
| 35  | 662  | 0        | 4.55e−13 | 0.0625  | 6.96e−13 | 0.0625  | 8.00e−11 |
| 36  | 675  | 0        | 1.86e−09 | 0.0625  | 1.68e−09 | 0.078125 | 2.30e−09 |
| 37  | 685  | 0 0.03125 | 5.46e−12 | 0.078125 | 1.64e−11 | 0.0625  | 1.46e−10 |
| 38  | 726  | 0        | 7.15e−07 | 0.1875  | 1.52e−06 | 0.21875 | 4.90e−13 |
| 39  | 729  | 0        | 9.31e−10 | 0.1875  | 2.51e−09 | 0.25    | 7.73e−07 |
| 40  | 817  | 0        | 4.69e−02 | 0.140625 | 2.81e−01 | 0.140625 | 2.28e−06 |
| 41  | 817  | 0        | 3.33e−09 | 0.0625  | 3.73e−09 | 0.078125 | 3.33e−16 |
| 42  | 957  | 0        | 1.53e−05 | 0.078125 | 3.81e−05 | 0.0625  | 2.27e−07 |
| 43  | 960  | 0        | 8.53e−14 | 0.234375 | 2.34e−13 | 0.171875 | 1.05e−11 |
| 44  | 1050 | 0        | 3.35e−08 | 0.140625 | 4.00e−08 | 0.125   | 1.13e+00 |
| 45  | 1074 | 0 0.03125 | 3.05e−05 | 0.069375 | 3.05e−05 | 0.069375 | 2.49e−11 |
| 46  | 1074 | 0        | 2.33e−10 | 0.09375 | 2.33e−10 | 0.078125 | 2.22e−16 |
| 47  | 1083 | 0 0.015625 | 1.49e−08 | 0.296875 | 2.97e−08 | 0.3125  | 3.07e−12 |
| 48  | 1083 | 0        | 5.42e−20 | 0.0625  | 5.42e−20 | 0.078125 | 2.22e−16 |
| 49  | 1086 | 0        | 7.45e−09 | 0.109375 | 1.02e−08 | 0.109375 | 5.73e−12 |
| 50  | 1138 | 0        | 1.64e−11 | 0.171875 | 1.37e−11 | 0.3125  | 4.20e−09 |
| 51  | 1224 | 0        | 4.66e−10 | 0.328125 | 6.99e−09 | 0.21875 | 1.90e−13 |
| 52  | 1282 | 0        | 9.31e−10 | 0.1875  | 1.05e−09 | 0.1875  | 4.59e−12 |
| 53  | 1440 | 0 0.015625 | 4.07e−10 | 0.328125 | 4.51e−10 | 0.296875 | 1.76e−13 |
| 54  | 1473 | 0        | 2.38e−07 | 0.359375 | 3.58e−07 | 0.296875 | 9.93e−11 |
| 55  | 1473 | 0 0.0625 | 2.38e−07 | 0.375  | 3.58e−07 | 0.4375  | 9.93e−11 |
| 56  | 1473 | 0        | 3.55e−15 | 0.09375 | 7.11e−15 | 0.078125 | 3.33e−16 |
| 57  | 1473 | 0        | 2.66e−15 | 0.234325 | 3.11e−15 | 0.21875 | 3.98e−13 |
| 58  | 1733 | 0 0.015625 | 1.56e−02 | 0.296875 | 3.05e−05 | 0.296875 | 2.62e−06 |
| 59  | 1806 | 0 0.03125 | 1.91e−06 | 0.8125  | 5.72e−06 | 0.796875 | 1.70e−11 |
| 60  | 1821 | 0 0.015625 | 2.79e−09 | 0.59375 | 4.25e−09 | 0.53125 | 1.10e−06 |
| 61  | 1824 | 0 0.03125 | 4.66e−09 | 0.826125 | 5.41e−09 | 0.859475 | 6.41e−11 |
| 62  | 1919 | 0 0.25   | 1.78e−15 | 4.984375 | 2.83e−15 | 5    | 2.21e+02 |
| 63  | 1922 | 0 0.03125 | 3.05e−05 | 0.671875 | 4.58e−05 | 0.5625  | 1.27e−10 |
| 64  | 1922 | 0        | 2.22e−16 | 0.125   | 2.22e−16 | 0.125   | 2.22e−16 |

$F1_t$: time of finding $\hat{x}$ for problem with $\hat{x}^T = \{1, \ldots, 1\}$.
On a conjugate directions method for solving strictly convex quadratic programming problems. The first stage generates a set of vectors mutually conjugate with respect to the hessian of the QP problem. It is followed by \( n \) independent directional minimizations executed concurrently from the same starting point \( \mathbf{x}^0 \). We think that it may be useful for distributed calculations in cloud computing.

Additionally, we find in that way a factorization of the inverse matrix \( \mathbf{G}^{-1} \) without calculating factorization of \( \mathbf{G} \) itself. It has the form \( \mathbf{RD}^{-1}\mathbf{R}^T \), where columns of matrix \( \mathbf{R} \) are the conjugate vectors found by the first stage algorithm. Knowledge of such inverse factorization may be very useful for problems both from optimization (when we have a family of QP problems with different linear terms) and linear algebra problems (with multiple right-hand sides). Our approach may be also used in conjunction with the Bientinesi et al. (2008) one sweep procedure to calculate the inverse of a
symmetric, positive definite matrix. It may be applied to find the factorization of the particular Schur block components.

The first stage is essentially sequential. But our second idea of applying \( n \) independent directional minimizations of the associated quadratic programming problem permits distribution of calculations in the second stage. It should be stressed that the solution accuracy of two algorithms tested in the second stage was exactly the same. All digits in the reported errors were equal.

There are still some open questions which require further investigation. The first one is, what are the classes of sparse problems, where the approach presented in the paper will be better than the standard factorization of \( G \) followed by the solution of two sets of linear equations with triangular matrices. Second is how to apply those ideas to the problems with the block structure. May be the fact that columns of factor \( L \) in the decompositions \( G = LDL^T \) or \( G = LL^T \) are mutually conjugate with respect to \( G^{-1} \) would serve as a hint. Another possibility is to combine the block approach of Bientinesi et al. (2008) with our approach applied to particular blocks. Factorization generated by means of our approach may be also used as the starting one in the refinement technique studied by Ogita and Oishi (2012) and Yanagisawa et al. (2014).

The reported computational results should be treated as preliminary. There are of course some issues which should be clarified. The most important are: identification of the suitable sparsity structure and selection of the appropriate values of parameter \( c \). We are not restricted to the choice of \( c = 1 \), which was used in our test calculations. This may be especially important in establishing the numerical stability of the method. To solve that task the most promising idea would be the normalization of the consecutive conjugate vectors.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

Alléon G, Carpentieri B, Duff IS, Giraud L, Langou J, Martin É, Sylvand G (2003) Efficient parallel iterative solvers for the solution of large dense linear systems arising from the boundary element method in electromagnetism. In: Proceedings of the international conference on supercomputing in nuclear application (SNA), Paris
Andrea C, Giovanni F, Massimo R (2017) Novel preconditioners based on quasi-Newton updates for nonlinear conjugate gradient methods. Opt Lett 11(4):835–853
Benoit C (1924) Note sur une méthode de résolution des équations normales provenant de l’application de la méthode des moindres carrés à un système d’équations lineaires en nombre inférieur à celui des inconnues (Procede du Commandant Cholesky). Bull Geod 2:67–77
Bertsekas DP (1995) Nonlinear programming. Athens Scientific, Belmont
Bertsekas DP, Tsitsiklis JN (1989) Parallel and distributed computation, numerical methods. Prentice Hall, Englewood Cliffs
Bientinesi P, Gunter B, van de Geijn RA (2008) Families of algorithms related to the inversion of a symmetric positive definite matrix. ACM Trans Math Softw 35(1):3
Boisvert RF, Pozo R, Remington K, Barrett R, Dongarra J, Miller B, Lipman B (1997a) The matrix market. http://math.nist.gov/MatrixMarket. Accessed October 2016
Boisvert RF, Pozo R, Remington K, Barrett R, Dongarra JJ (1997b) The matrix market: a web resource for test matrix collections. In: Boisvert RF (ed) Quality of numerical software, assessment and enhancement. Chapman & Hall, London, pp 125–137
Chen X, Zhou W (2010) Smoothing nonlinear conjugate gradient method for image restoration using nonsmooth nonconvex minimization. SIAM J Imaging Sci 3(4):765–790
Dai YH (2011) Nonlinear conjugate gradient methods. ftp://lseec.cc.ac.cn/pub/dyh/papers/CGoverview.pdf. Accessed 26 May 2017
Dai YH, Yuan Y (1999) A nonlinear conjugate gradient method with a strong global convergence property. SIAM J Optim 10(1):177–182
Davis TA (2014) University of Florida sparse matrix collection. University of Florida, Gainesville
Davis TA, Hu Y (2011a) The University of Florida sparse matrix collection. ACM Trans Math Softw (TOMS) 38(1), article 1 http://www.cise.ufl.edu/research/sparse/matrixs/. Accessed October 2016
Davis TA, Hu Y (2016) The University of Florida sparse matrix collection. http://yifanhu.net/PUB/matrices.pdf. Accessed October 2016
Davis TA, Rajamanickam S, Sid-Lakhdar WM (2016) A survey of direct methods for sparse linear systems. Technical report, Department of Computer Science and Engineering, Texas A&M University. http://faculty.cse.tamu.edu/davis/publications.html. Accessed May 2017
Duff IS, Grimes RG, Lewis JG (1989a) Harwell/Boeing collection. http://math.nist.gov/MatrixMarket/collections/hb.html. Accessed October 2016
Duff IS, Grimes RG, Lewis JG (1997) The Rutherford–Boeing sparse matrix collection. Tech report RAL-TR-97-031, Rutherford Appleton Laboratory, Oxford, UK
Duff IS, Grimes RG, Lewis JG (1989b) Sparse matrix test problems. ACM Trans Math Softw 15(1):1–14
El Guennouni A, Jbilou K, Sadok H (2003) A block version of BIGSTAB for linear systems with multiple right-hand sides. ETNA El Trans Numer Anal 16:129-142. https://eudml.org/doc/124803
Fasano G, Roma M (2016) A novel class of approximate inverse preconditioners for large positive definite linear systems in optimization. Comp Optim Appl 65(2):399–429
Felippa CA (2001) A historical outline of matrix structural analysis: a play in three acts. Comp Struct 79(14):1313–1324, doi:10.1016/S0045-7949(01)00025-6
Feng YT, Owen DRJ, Peri D (1995) A block conjugate gradient method applied to linear systems with multiple right-hand sides. Comp Meth Appl Mech Eng 127(1-4):203–215
Fletcher R (1987) Practical methods of optimization, sec edn. Wiley, Chichester
Fletcher R, Reeves CM (1964) Function minimization by conjugate gradients. Comput J 7(2):149–154
Gratton S, Sartenaer A, Ilunga JT (2011) On a class of limited memory preconditioners for large scale linear systems with multiple right-hand sides. SIAM J Optim 21(3):912–935
Hager WW, Zhang H (2005) A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J Optim 16(1):170–192
Hestenes MR, Stiefel EL (1952) Methods of conjugate gradients for solving linear systems. J Res Nat Bur Stand 49(6):409–436
Inverse problems. In: Wikipedia. https://en.wikipedia.org/wiki/Inverse_problem. Accessed May 2017
Lin L, Yang C, Meza JC, Lu J, Ying L (2011) SelInv-an algorithm for selected inversion of a sparse symmetric matrix. J ACM Trans Math Softw (TOMS) 37(4):40. doi:10.1145/1916461.1916464
McGuire W, Gallagher RH, Ziemian RD (2000) Matrix structural analysis, 2nd edn. Wiley, New York
Ogita T, Oishi S (2012) Accurate and robust inverse Cholesky factorization. Nonlinear Theory Appl IEICE 3(1):103–111
O’Leary DP (2000) Symbiosis between linear algebra and optimization. J Comp Appl Math 123(1–2):447–465
Polak E, Ribière G (1969) Note sur la convergence de méthodes de directions conjuguées. Rev Francaise Inf Rech Opér 3(16):35–43
Polyak BT (1969) The conjugate gradient method in extreme problems. USSR Comp Math Math Phys 9(4):94–112
Powell MJD (1964) An efficient method for finding the minimum of a function of several variables without calculating derivatives. Comp J 7(2):155–162
Simoncini V, Szyld D (2007) Recent computational developments in Krylov subspace methods for linear systems. Numer Linear Algebra Appl 14(1):1–59
Stachurski A (2012) Quadratic approximations in nonlinear continuous optimization (in polish: Aproksymacje kwadratowe w ciągłej optymalizacji nieliniowej). Prace Naukowe Politechniki Warszawskiej, Elektronika 184, Publishing House of WUT, Warsaw
Stoer J (1972) 1976) Einfuehrung in die numerische mathematik I. Springer, Berlin
Stoer J, Bulirsch R (1993) Introduction to numerical analysis. Springer, Berlin
Wilkinson JH (1963) Rounding errors in algebraic processes. Prentice-Hall Inc., Englewood Cliffs
Yanagisawa Y, Ogita T, Oishi S (2014) Convergence analysis of an algorithm for accurate inverse Cholesky factorization. Jpn J Ind Appl Math 31(3):461–482. doi: 10.1007/s13160-014-0154-4
Yunlan F, Huirao Z, Wuhan XH (1997) Fast inverse Cholesky decomposition for rectangular Toeplitz-block matrix. Univ J Nat Sci 2(4):418–422. doi: 10.1007/BF02830245
Zhu H, Chen W, Li B, Gao F (2011) An improved square-root algorithm for V-BLAST based on efficient inverse Cholesky factorization. IEEE Trans Wireless Commun 10(1):43–48