Low Rank Updates for the Cholesky Decomposition

Matthias Seeger
Department of EECS
University of California at Berkeley
485 Soda Hall, Berkeley CA
mseeger@cs.berkeley.edu

April 5, 2008

Abstract

Usage of the Sherman-Morrison-Woodbury formula to update linear systems after low rank modifications of the system matrix is widespread in machine learning. However, it is well known that this formula can lead to serious instabilities in the presence of roundoff error. If the system matrix is symmetric positive definite, it is almost always possible to use a representation based on the Cholesky decomposition which renders the same results (in exact arithmetic) at the same or less operational cost, but typically is much more numerically stable. In this note, we show how the Cholesky decomposition can be updated to incorporate low rank additions or downdated for low rank subtractions. We also discuss a special case of an indefinite update of rank two. The methods discussed here are well-known in the numerical mathematics literature, and code for most of them can be found in the `LINPACK` suite.

Note: Matlab MEX implementations for most of the techniques described here are available for download at *http://www.kyb.tuebingen.mpg.de/bs/people/seeger/*. If you make use of them (subject to the license), you have to cite this report and the website for obtaining the code in your publications.

1 The Problem and the Primitives

Let $\boldsymbol{A} \in \mathbb{R}^{n,n}$ be symmetric positive definite (we write $\boldsymbol{A} \succ \boldsymbol{0}$), i.e. $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \neq \boldsymbol{0}$. Then, $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$ for a lower triangular matrix $\boldsymbol{L}$ with positive diagonal elements, and this *Cholesky decomposition* is unique.

Forget about ever inverting $\boldsymbol{A}$ if you don't have to! Almost everything which you might want $\boldsymbol{A}^{-1}$ for can be done equally fast or faster using $\boldsymbol{L}$, and will usually be much more numerically stable on a computer. The system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is solved as $\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}$, $\boldsymbol{L}^T\boldsymbol{x} = \boldsymbol{y}$ which needs two *back-substitutions* (the procedure of solving a system with a triangular matrix).[1] The operation count for a back-substitution is about half than for a matrix-vector multiplication, so even if $\boldsymbol{A}^{-1}$ was given exactly there would be no gain in efficiency. However, the way via the Cholesky factorization in general leads to a much more accurate

[1]Confusingly (and historically) this procedure is termed differently depending on whether the system matrix is upper or lower triangular (forward- and back-substitution), although one is obtained from the other simply by inverting the vector index. We do not follow this nomenclature to avoid unnecessary confusion.

solution for $\boldsymbol{x}$. In fact, the best general way of inverting $\boldsymbol{A}$ would be to compute $\boldsymbol{L}$ and then $\boldsymbol{A}^{-1}$ using an algorithm based on repeated back-substitutions.

Frequently occuring terms are computed as $\boldsymbol{x}^T\boldsymbol{A}^{-1}\boldsymbol{x} = \|\boldsymbol{L}^{-1}\boldsymbol{x}\|^2$, $\log|\boldsymbol{A}| = 2\log|\boldsymbol{L}| = 2\boldsymbol{1}^T\log(\operatorname{diag}\boldsymbol{L})$. $\operatorname{tr}\boldsymbol{A}^{-1}$ can be computed from $\boldsymbol{L}$ in about half the time than $\boldsymbol{A}^{-1}$.[2]

Suppose now for a given statistical problem we use a representation based on a Cholesky decomposition $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$. By a "representation" we mean that we store $\boldsymbol{L}$ explicitly in order to do back-substitutions on demand, or we maintain $\boldsymbol{X} = \boldsymbol{L}^{-1}\boldsymbol{B}$ for some fixed matrix $\boldsymbol{B} \in \mathbb{R}^{n,m}$, or both. In the following we concentrate on this situation, although it can easily be extended to incorporate more involved usages of $\boldsymbol{L}$.

One of the simplest modifications is adding a new row/column to $\boldsymbol{A}$. The corresponding update of $\boldsymbol{L}$ and $\boldsymbol{X}$ is fairly obvious and will not be discussed here. In this paper, we are interested in updating the representation if $\boldsymbol{A}$ is modified by adding a symmetric low rank matrix. In the simplest case, $\boldsymbol{A}' = \boldsymbol{A} + \boldsymbol{v}\boldsymbol{v}^T$ (*positive rank-1*, *update*) or $\boldsymbol{A}' = \boldsymbol{A} - \boldsymbol{v}\boldsymbol{v}^T$ (*negative rank-1*, *downdate*). These cases are fundamentally different in that for a positive update $\boldsymbol{A}' \succeq \boldsymbol{A}$, as long as $\boldsymbol{A}$ is well-conditioned, our procedure will not run into trouble and $\boldsymbol{A}'$ has larger eigenvalues than $\boldsymbol{A}$.[3] On the other hand, negative updates break down if $\boldsymbol{A}'$ is not positive definite and can result in large errors if $\boldsymbol{A}'$ is close to singularity. Low rank updates of the form $\boldsymbol{A}' = \boldsymbol{A} \pm \boldsymbol{V}\boldsymbol{V}^T$, $\boldsymbol{V} \in \mathbb{R}^{n,d}$ can always be done by applying d updates/downdates sequentially.

In case of an indefinite low rank update, we do not know of a method which can be recommended in general. However, stable methods are available for special forms of indefinite rank two updates and may apply in cases of interest. Important examples are given in Section 4 and Section 5.

In the remainder of this note, we assume that the factor $\boldsymbol{L}$ is given explicitly, and is to be overwritten by the new factor $\boldsymbol{L}'$. In some applications, one is interested rather in an $O(n)$ representation of $\tilde{\boldsymbol{L}}$ s.t. $\boldsymbol{L}\tilde{\boldsymbol{L}} = \boldsymbol{L}'$. We comment on this issue briefly in each case, the reader will have not problem filling in the details.

2 Rank One Update

Several methods for updating or downdating a Cholesky factor after a modification of rank one have been proposed. A review is given in [2]. Note that the same techniques can be used to update a QR decomposition. Maybe the most stable techniques have been proposed by Stewart, and code for them can be found[4] in `LINPACK` [1]. Here, we discuss the `dchud` routine for $\boldsymbol{A}' = \boldsymbol{A} + \boldsymbol{v}\boldsymbol{v}^T$.

In this section we make use of matrices and vectors of size $n+1$ which are indexed as $0, \dots, n$ for convenience. A *Givens* (or plane) rotation applied to $\boldsymbol{x} \in \mathbb{R}^{n+1}$ rotates two components of $\boldsymbol{x}$ by some angle θ and leaves all other components the same. Namely,

[2]It can also be approximated using the *randomized trace technique* which uses $\operatorname{tr}\boldsymbol{A}^{-1} = \mathrm{E}[\boldsymbol{x}^T\boldsymbol{A}^{-1}\boldsymbol{x}]$, $\boldsymbol{x} \sim N(\boldsymbol{0}, \boldsymbol{I})$ together with the law of large numbers.

[3]See the interlacing theorems in [4].

[4]`LINPACK` is superseded by the much more efficient `LAPACK` suite today, but the Cholesky updating/downdating routines are not yet in there, although their addition is planned (pers. comm.).

$\boldsymbol{J}_k = \boldsymbol{I} + (c_k - 1)(\boldsymbol{\delta}_0\boldsymbol{\delta}_0^T + \boldsymbol{\delta}_k\boldsymbol{\delta}_k^T) + s_k(\boldsymbol{\delta}_0\boldsymbol{\delta}_k^T - \boldsymbol{\delta}_k\boldsymbol{\delta}_0^T)$, where $c_k = \cos\theta_k, s_k = \sin\theta_k$. Stewarts ansatz is

$$\boldsymbol{J}_n \dots \boldsymbol{J}_1 \left[\boldsymbol{v}, \boldsymbol{L}\right]^T = \left[\boldsymbol{0}, \boldsymbol{L}'\right]^T, \tag{1}$$

leading to $\boldsymbol{L}'\boldsymbol{L}'^T = \boldsymbol{L}\boldsymbol{L}^T + \boldsymbol{v}\boldsymbol{v}^T$.

The algorithm proceeds $k = 1, \dots, n$. $\boldsymbol{J}_k$ is chosen based on the k-th column, namely $\boldsymbol{J}_k[v_k, \boldsymbol{L}_{k,k}]^T = [0, \boldsymbol{L}'_{k,k}]^T$. The `BLAS` routine `drotg` can be used to compute c_k, s_k in a stable way. If this results in $\boldsymbol{L}'_{k,k} < 0$, we simply flip $c_k \leftarrow -c_k$, $s_k \leftarrow -s_k$, and $\boldsymbol{L}'_{k,k} \leftarrow -\boldsymbol{L}'_{k,k}$. $\boldsymbol{v}_{>k}$ and $\boldsymbol{L}_{>k,k}$ are updated by applying the rotation $\boldsymbol{J}_k$, the latter forms $\boldsymbol{L}'_{>k,k}$. The `BLAS` routine `drot` can be used for this plane rotation.

The update factor $\tilde{\boldsymbol{L}}$ is given implicitly by $\{(c_k, s_k)\}$, in that $\tilde{\boldsymbol{L}}\boldsymbol{x} = \boldsymbol{y}$ iff $[0, \boldsymbol{y}^T] = [*, \boldsymbol{x}^T]\boldsymbol{J}_n \dots \boldsymbol{J}_1$.

We can also "drag along" solutions, by which we mean updating $\boldsymbol{Z} \to \boldsymbol{Z}'$ s.t. $\boldsymbol{L}\boldsymbol{Z} + \boldsymbol{v}\boldsymbol{y}^T = \boldsymbol{L}'\boldsymbol{Z}'$. To see how this works, append $[\boldsymbol{y}, \boldsymbol{Z}^T]^T$ to the l.h.s., $[\boldsymbol{\zeta}, \boldsymbol{Z}'^T]^T$ to the r.h.s. in Eq. 1. We initialize $\boldsymbol{\zeta} = \boldsymbol{y}$, then iterate $k = 1, \dots, n$. In the k-th step, $[\boldsymbol{\zeta}, \boldsymbol{Z}'^T_{k,\cdot}] \leftarrow [\boldsymbol{\zeta}, \boldsymbol{Z}^T_{k,\cdot}]\boldsymbol{J}_k^T$, which can be done by `drot`.

Note that the `LINPACK` routine `dchud` does not make good use of `BLAS`, but rather contains explicit $O(n^2)$ loops. Our implementation uses `drot` whenever possible, which can be much faster, but we require a scratch vector of size $\max\{n, r\}$ if $\boldsymbol{Z} \in \mathbb{R}^{n,r}$. Another slight problem with `dchud` is that it can result with negative elements on $\operatorname{diag}\boldsymbol{L}'$. This is not contrary to its specification, but somewhat non-standard. In our implementation, $\operatorname{diag}\boldsymbol{L}' \succ \boldsymbol{0}$.

3 Rank One Downdate

As noted above, a Cholesky downdate, namely $\boldsymbol{L} \to \boldsymbol{L}'$ if $\boldsymbol{A}' = \boldsymbol{A} - \boldsymbol{v}\boldsymbol{v}^T$, is more difficult to do in a numerically stable manner. Stewart provides `dchdd` in `LINPACK`, which we discuss here. Again, our implementation uses `BLAS` routines whenever possible, while `dchdd` contains explicit $O(n^2)$ loops.

Again, we extend objects to size $n+1$ with an index 0, and use Givens rotations $\boldsymbol{J}_k$. Here, the ansatz is

$$\boldsymbol{J}_1 \dots \boldsymbol{J}_n \left[\boldsymbol{0}, \boldsymbol{L}\right]^T = \left[\boldsymbol{v}, \boldsymbol{L}'\right]^T, \tag{2}$$

leading to $\boldsymbol{L}'\boldsymbol{L}'^T = \boldsymbol{L}\boldsymbol{L}^T - \boldsymbol{v}\boldsymbol{v}^T$. By left-multiplication with $\boldsymbol{\delta}_0^T$, we obtain that if

$$\boldsymbol{L}\boldsymbol{p} = \boldsymbol{v}, \quad \rho^2 = 1 - \boldsymbol{p}^T\boldsymbol{p}, \quad \boldsymbol{q} = [\rho, \boldsymbol{p}^T]^T,$$

then $\boldsymbol{J}_1 \dots \boldsymbol{J}_n\boldsymbol{q} = \boldsymbol{\delta}_0$ (we use the fact that $\|\boldsymbol{q}\| = 1$ as a rotation of $\boldsymbol{\delta}_0$ in order to determine ρ). Note that $\boldsymbol{L}\boldsymbol{L}^T - \boldsymbol{v}\boldsymbol{v}^T$ is positive definite iff the expression for ρ^2 is positive. The method breaks down if this is not the case. We now iterate $k = n, \dots, 1$. $\boldsymbol{J}_k$ is chosen to annulate the k-th element of $\boldsymbol{q}$. c_k, s_k can be computed by `BLAS` `drotg`. Note that in this process, the value of q_0 is nondecreasing (if `drotg` flips it to a negative value, we compensate for that by flipping signs of c_k, s_k), and should eventually attain 1 (up to roundoff). Since (in exact arithmetic) $c_k = q_0/\sqrt{q_0^2 + q_k^2}$ in iteration k, we know that $c_k \geq (1 + (q_k/\rho)^2)^{-1/2}$, so is bounded away from zero. This will be important in a moment.

Given the c_k, s_k, we can update $\boldsymbol{L} \to \boldsymbol{L}'$ in a similar way as in Section 2. However, in this case we do not see how one could compensate for $\boldsymbol{L}'_{k,k} < 0$ by simply flipping the values c_k, s_k. In order to obtain $\operatorname{diag} \boldsymbol{L}' \succ \boldsymbol{0}$, we write

$$\boldsymbol{L}' = \boldsymbol{L}'_{[chdd]}(\operatorname{diag} \boldsymbol{d}), \quad d_i \in \{-1, +1\}.$$

Here, $\boldsymbol{L}'_{[chdd]}$ is the factor which we obtained without consideration of $\operatorname{diag} \boldsymbol{L}'$, which is what `dchdd` does.

"Dragging along", *i.e.* $\boldsymbol{L}'\boldsymbol{Z}' = \boldsymbol{L}\boldsymbol{Z} - \boldsymbol{v}\boldsymbol{y}^T$, works by appending $[\boldsymbol{\zeta}, \boldsymbol{Z}^T]^T$ to the l.h.s., $[\boldsymbol{y}, \boldsymbol{Z}'^T]^T$ to the r.h.s. of Eq. 2. We initialize $\boldsymbol{\zeta} = \boldsymbol{y}$, then iterate $k = 1, \dots, n$. In the k-th iteration, $c_k \boldsymbol{Z}'_{k,\cdot} = \boldsymbol{Z}_{k,\cdot} - s_k \boldsymbol{\zeta}^T$, and $\boldsymbol{\zeta}^T \leftarrow c_k \boldsymbol{\zeta}^T - s_k \boldsymbol{Z}'_{k,\cdot}$. We need to divide through c_k here, which is stable since they are bounded away from zero. As opposed to the situation for an update, we cannot simply use `drot` here, but need several `BLAS` calls. We end up with $\boldsymbol{L}'_{[chdd]}\boldsymbol{Z}' = \boldsymbol{L}\boldsymbol{Z} - \boldsymbol{v}\boldsymbol{y}^T$, therefore need to replace $\boldsymbol{Z}' \leftarrow (\operatorname{diag} \boldsymbol{d})\boldsymbol{Z}'$ still.

Why does this work? For $k \in \{1, \dots, n\}$, let $[\boldsymbol{\zeta}^{(k-1)}, \boldsymbol{M}^T]^T = \boldsymbol{J}_{k-1}^T \dots \boldsymbol{J}_1^T [\boldsymbol{y}, \boldsymbol{Z}'^T]^T$ (here and elsewhere, empty matrix products equate to $\boldsymbol{I}$). Since $\boldsymbol{J}_j$ operates on dimensions 0 and j only, we have that $\boldsymbol{M}_{k,\cdot} = \boldsymbol{Z}'_{k,\cdot}$. Multiplying the equation for $\boldsymbol{\zeta}^{(k)}$ by $\boldsymbol{\delta}_0^T$, we obtain $\boldsymbol{\zeta}^{(k)T} = \boldsymbol{\delta}_0^T \boldsymbol{J}_k^T [\boldsymbol{\zeta}^{(k-1)}, \boldsymbol{M}^T]^T = [c_k, -s_k][\boldsymbol{\zeta}^{(k-1)}, \boldsymbol{Z}'_{k,\cdot}{}^T]^T$. The l.h.s. of this equation can also be written as $\boldsymbol{J}_{k+1} \dots \boldsymbol{J}_n [\boldsymbol{\zeta}^{(n)}, \boldsymbol{Z}^T]^T$, whose k-th row is just $\boldsymbol{Z}_{k,\cdot}$. Multiplying the equation by $\boldsymbol{\delta}_k^T$, we obtain $\boldsymbol{Z}_{k,\cdot} = [s_k, c_k][\boldsymbol{\zeta}^{(k-1)}, \boldsymbol{Z}'_{k,\cdot}{}^T]^T$. These are the update equations above, where $\boldsymbol{\zeta}$ contains the successive $\boldsymbol{\zeta}^{(k)}$.

We also need to compute $\boldsymbol{p}$, which can be done by `LAPACK dtrsv`. We require a scratch vector of size $\max\{n, r\}$ if $\boldsymbol{Z} \in \mathbb{R}^{n,r}$.

The update factor $\tilde{\boldsymbol{L}}$ is given implicitly by $\{(c_k, s_k)\}$ and $\boldsymbol{d}$, in that $\tilde{\boldsymbol{L}}\boldsymbol{x} = \boldsymbol{y}$ iff $[0, (\boldsymbol{d} \circ \boldsymbol{x})^T] = [*, \boldsymbol{y}^T]\boldsymbol{J}_n^T \dots \boldsymbol{J}_1^T$.

4 Indefinite Rank Two Update: An Examples

Attention: As for this moment, we do not have a working implementation of Goldfarb's method [3], our code does not work for reasons unclear to us. We have not found this method being implemented in `LINPACK`. Therefore, we cannot recommend the method to be described in this section from our own experiences. If the reader manages to obtain a correct implementation, or spots a mistake in our description here, we would be very grateful for a note.

Suppose $\boldsymbol{A}' = \boldsymbol{A} + \boldsymbol{B}$ where $\boldsymbol{A}'$ is known to be positive definite in exact arithmetic, and $\boldsymbol{B}$ is symmetric low rank. We have discussed special cases where $\boldsymbol{B}$ is positive or negative semidefinite, but we do not known of a method which can be recommended in general if $\boldsymbol{B}$ is indefinite (*i.e.* has positive and negative eigenvalues). A general idea would be to apply rank one updates (as described above) sequentially, but the ordering becomes very important (due to the presence of updates *and* downdates), and it is not clear how to select one which leads to a stable update.

In this section we concentrate on the case $\operatorname{rk} \boldsymbol{B} = 2$. Stable methods for indefinite rank two updates of special form have been suggested. Faced with a particular $\boldsymbol{B}$ (of rank two), we recommend trying to reduce it to a sequence of stable rank two and positive rank one updates. In this section we give a concrete example.

Let $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$. Goldfarb [3] gives a stable method for indefinite rank two updates of the form

$$\boldsymbol{A}' = \left(\boldsymbol{I} + \boldsymbol{v}\boldsymbol{u}^T\right) \boldsymbol{A} \left(\boldsymbol{I} + \boldsymbol{u}\boldsymbol{v}^T\right).$$

The method is based on an orthonormal triangularization of the matrix $\boldsymbol{I} + \boldsymbol{z}\boldsymbol{w}^T$, *i.e.* $(\boldsymbol{I}+\boldsymbol{z}\boldsymbol{w}^T)\boldsymbol{Q} = \tilde{\boldsymbol{L}}$, where $\boldsymbol{Q}$ is orthonormal and $\tilde{\boldsymbol{L}}$ is lower triangular. Now, if $\boldsymbol{L}\boldsymbol{z} = \boldsymbol{v}$, $\boldsymbol{w} = \boldsymbol{L}^T\boldsymbol{u}$, then

$$\boldsymbol{A}' = \boldsymbol{L}\left(\boldsymbol{I} + \boldsymbol{z}\boldsymbol{w}^T\right)\left(\boldsymbol{I} + \boldsymbol{w}\boldsymbol{z}^T\right)\boldsymbol{L}^T = \boldsymbol{L}\tilde{\boldsymbol{L}}\boldsymbol{Q}^T\boldsymbol{Q}\tilde{\boldsymbol{L}}^T\boldsymbol{L}^T = \boldsymbol{L}'\boldsymbol{L}'^T, \quad \boldsymbol{L}' = \boldsymbol{L}\tilde{\boldsymbol{L}},$$

because $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$. Furthermore, $\tilde{\boldsymbol{L}}$ has a special form

$$\tilde{\boldsymbol{L}} = \operatorname{diag}\boldsymbol{\lambda} + \left[\boldsymbol{w}\boldsymbol{\beta}^T + \boldsymbol{z}\boldsymbol{\gamma}^T\right]_{LT},$$

where $[\boldsymbol{B}]_{LT} := (b_{ij}\mathrm{I}_{\{i>j\}})_{ij}$. Thus, $\tilde{\boldsymbol{L}}$ is given in terms of $O(n)$ parameters $\boldsymbol{\lambda} \in \mathbb{R}^n$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$, $\boldsymbol{w}_{>1}$, $\boldsymbol{z}_{>1} \in \mathbb{R}^{n-1}$. The computation of these parameters is $O(n)$ once $\boldsymbol{w}$, $\boldsymbol{z}$ are given. The explicit update $\boldsymbol{L} \to \boldsymbol{L}'$ can be done in place in $O(n^2)$, as shown in Algorithm 1. Back-substitution with $\tilde{\boldsymbol{L}}$ is $O(n)$ and is given in Algorithm 2.

Algorithm 1 Update of Cholesky Factor, Indefinite Rank-2 Update.

```
for $i = 1, \dots, n$ do
    $\rho = l_{i,i}$. $l'_{i,i} = l_{i,i}\lambda_i$. $\sigma_\beta = \sigma_\gamma = 0$.
    for $j = i-1, i-2, \dots, 1$ do
        $\sigma_\beta \leftarrow \sigma_\beta + \rho w_{j+1}$, $\sigma_\gamma \leftarrow \sigma_\gamma + \rho z_{j+1}$.
        $\rho = l_{i,j}$. $l'_{i,j} = l_{i,j}\lambda_j + \sigma_\beta\beta_j + \sigma_\gamma\gamma_j$.
    end for
end for
```

Algorithm 2 Back-substitution $\boldsymbol{x} = \tilde{\boldsymbol{L}}^{-1}\boldsymbol{b}$, Indefinite Rank-2 Update.

```
$x_1 = b_1/\lambda_1$, $\sigma_\beta = \sigma_\gamma = 0$.
for $i = 2, \dots, n$ do
    $\sigma_\beta \leftarrow \sigma_\beta + \beta_{i-1}x_{i-1}$, $\sigma_\gamma \leftarrow \sigma_\gamma + \gamma_{i-1}x_{i-1}$.
    $x_i = \lambda_i^{-1}(b_i - \sigma_\beta w_i - \sigma_\gamma z_i)$.
end for
```

Note that in this context, the following modification of $\boldsymbol{X}$ may be equally useful. Recall that $\boldsymbol{L}\boldsymbol{X} = \boldsymbol{B}$. Suppose $\boldsymbol{X}'$ is required s.t. $\boldsymbol{L}'\boldsymbol{X}' = (\boldsymbol{I} + \boldsymbol{v}\boldsymbol{u}^T)\boldsymbol{B}$. Note that Goldfarb's method determines a orthonormal $\boldsymbol{Q}$ s.t. $(\boldsymbol{I} + \boldsymbol{v}\boldsymbol{u}^T)\boldsymbol{L}\boldsymbol{Q} = \boldsymbol{L}'$. Therefore, if $\boldsymbol{X}' = \boldsymbol{Q}^T\boldsymbol{X}$, then

$$\boldsymbol{L}'\boldsymbol{X}' = (\boldsymbol{I} + \boldsymbol{v}\boldsymbol{u}^T)\boldsymbol{L}\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{X} = (\boldsymbol{I} + \boldsymbol{v}\boldsymbol{u}^T)\boldsymbol{B},$$

as desired. Since $\boldsymbol{Q}^T$ is a product of Givens rotations, all we need to do is to apply these to $\boldsymbol{X}$.

In the first example, suppose we have $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$, and we want to downdate $\boldsymbol{L} \to \boldsymbol{L}'$ according to the removal of column and row i in $\boldsymbol{A}$. If $i = n$, all we have to do is to remove the last row and column of $\boldsymbol{L}$. If $i \neq n$, let $\boldsymbol{A}'$ by obtained from $\boldsymbol{A}$ by exchanging rows/columns i and n. We can write

$$\boldsymbol{A}' = \boldsymbol{P}\boldsymbol{A}\boldsymbol{P}, \quad \boldsymbol{P} = \boldsymbol{I} - (\boldsymbol{\delta}_i - \boldsymbol{\delta}_n)(\boldsymbol{\delta}_i - \boldsymbol{\delta}_n)^T.$$

This means that we can use the method of Goldfarb with $\boldsymbol{u} = \boldsymbol{\delta}_i - \boldsymbol{\delta}_n$ and $\boldsymbol{v} = -\boldsymbol{u}$. We then obtain $\boldsymbol{L}'$ by removing the last row and column.

For the second example, suppose we have the following problem. Let

$$\boldsymbol{A} = \tilde{\boldsymbol{A}} + \boldsymbol{I}_{\cdot,I}\bar{\boldsymbol{A}}_I\boldsymbol{I}_{I,\cdot}$$

which depends on $I \subset \{1,\dots,n\}$. Here, $\tilde{\boldsymbol{A}} \succ \boldsymbol{0}$ and $\bar{\boldsymbol{A}} \succeq \boldsymbol{0}$. Let $I' = I \cup \{j\}$, $j \notin I$, and we need to do a stable update of the Cholesky decomposition $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$. We have

$$\boldsymbol{A}' = \boldsymbol{A} + 2\,\mathrm{sym}\,\boldsymbol{I}_{\cdot,I}\bar{\boldsymbol{A}}_{I,j}\boldsymbol{\delta}_j^T + \bar{\boldsymbol{A}}_j\boldsymbol{\delta}_j\boldsymbol{\delta}_j^T.$$

In order to employ Goldfarb's method, let $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{I}_{\cdot,I}\bar{\boldsymbol{A}}_{I,j}$ and $\boldsymbol{v} = \boldsymbol{\delta}_j$ (*i.e.* $\boldsymbol{w} = \boldsymbol{L}^T\boldsymbol{u} = \boldsymbol{L}^{-1}\boldsymbol{I}_{\cdot,I}\bar{\boldsymbol{A}}_{I,j}$). Then,

$$\boldsymbol{u}^T\boldsymbol{A}\boldsymbol{u} = \bar{\boldsymbol{A}}_{j,I}\boldsymbol{I}_{I,\cdot}\boldsymbol{A}^{-1}\boldsymbol{I}_{\cdot,I}\bar{\boldsymbol{A}}_{I,j}.$$

If

$$\alpha = \bar{\boldsymbol{A}}_j - \bar{\boldsymbol{A}}_{j,I}\boldsymbol{I}_{I,\cdot}\boldsymbol{A}^{-1}\boldsymbol{I}_{\cdot,I}\bar{\boldsymbol{A}}_{I,j},$$

we have

$$\boldsymbol{A}' = \left(\boldsymbol{I} + \boldsymbol{v}\boldsymbol{u}^T\right)\boldsymbol{A}\left(\boldsymbol{I} + \boldsymbol{u}\boldsymbol{v}^T\right) + \alpha\boldsymbol{v}\boldsymbol{v}^T.$$

Thus, if $\alpha \geq 0$, we can do a Goldfarb update followed by a positive rank one update. We give a proof in the following.

Note that $\cdot$ is a shortcut for $\{1,\dots,n\}$. To make that specific, let $J = \{1,\dots,n\}$ and $J' = \{1,\dots,n+1\}$. Extend the matrix $\bar{\boldsymbol{A}}$ to J' by imagining it being a kernel matrix over n points and duplicating point j as point $n+1$. In other words, $\bar{\boldsymbol{A}}_{i,j} = \bar{\boldsymbol{A}}_{i,n+1}$, $i = 1,\dots,n$, and $\bar{\boldsymbol{A}}_{n+1,n+1} = \bar{\boldsymbol{A}}_{j,j}$. The extended matrix is positive semidefinite as well. Let $I'' = I \cup \{n+1\}$. Now consider

$$\boldsymbol{M} = \boldsymbol{I}_{J',J}\tilde{\boldsymbol{A}}\boldsymbol{I}_{J,J'} + \boldsymbol{I}_{J',I''}\bar{\boldsymbol{A}}_{I''}\boldsymbol{I}_{I'',J'}$$

which is positive semidefinite as a sum of two positive semidefinite matrices. In fact, $\boldsymbol{M}_J = \boldsymbol{A}$ is positive definite, furthermore $\boldsymbol{M}_{J,n+1} = \boldsymbol{I}_{J,I}\bar{\boldsymbol{A}}_{I,n+1} = \boldsymbol{I}_{\cdot,I}\bar{\boldsymbol{A}}_{I,j}$, and $\boldsymbol{M}_{n+1,n+1} = \bar{\boldsymbol{A}}_{n+1,n+1} = \bar{\boldsymbol{A}}_{j,j}$. Then, we have $\alpha \geq 0$ by looking at the Schur complement of $n+1$ versus $\{1,\dots,n\}$ and the positive semidefiniteness of $\boldsymbol{M}$.

5 LINPACK DCHEX

In the context of specific low rank updates of a Cholesky factor, the `LINPACK` routine `dchex` is useful. Just as in `LAPACK`, the leading "d" stands for real double precision. `LINPACK` supports upper triangular Cholesky factors only, so we will follow this convention here.

Suppose that $\boldsymbol{A} = \boldsymbol{R}^T\boldsymbol{R}$, and that $\boldsymbol{A}' = \boldsymbol{E}'\boldsymbol{A}\boldsymbol{E}$ for a special permutation matrix $\boldsymbol{E}$. In fact, $\boldsymbol{E}$ is determined by $1 \leq k < l \leq n$ and job $\in \{1,2\}$. Depending on job, the columns in $\boldsymbol{A}\boldsymbol{E}$ are permuted as follows. If job $= 1$, they have the new ordering $1,\dots,k-1,l,k,\dots,l-$

$1, l+1, \dots, n$. If job $= 1$, the new ordering is $1, \dots, k-1, k+1, \dots, l, k, l+1, \dots, n$. Note that this is what $\boldsymbol{E}^T$ is doing to the coordinates of a vector.

Note that `dchex` solves a special case of Goldfarb's update, however it is faster and less prone to numerical problems. It does not require backsubstitutions with $\boldsymbol{R}^T$ in order to work. The method determines an orthonormal $\boldsymbol{U}$ s.t. $\boldsymbol{URE} = \boldsymbol{R}'$. Here, $\boldsymbol{U}$ is the product of $l-k$ Givens rotations. Optionally, for a given $\boldsymbol{X}$ the routine computes $\boldsymbol{X}' = \boldsymbol{UX}$. This is useful in the context of our requirements, because if $\boldsymbol{R}^T\boldsymbol{X} = \boldsymbol{B}$, then

$$\boldsymbol{R}'^T\boldsymbol{X}' = \boldsymbol{E}^T\boldsymbol{R}^T\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{X} = \boldsymbol{E}^T\boldsymbol{B}.$$

The problem of downdating a Cholesky factor after removal of row and column i of $\boldsymbol{A}$ has been mentioned above. We can use `dchex` with job $= 2$, $k = i$, $l = n$. After that, we simply drop the last row and column of $\boldsymbol{R}$.

Note: There is a bug in `dchex`. Namely, in some cases it returns $\boldsymbol{R}'$ with $\boldsymbol{R}'_{l,l} < 0$, in fact the whole row $\boldsymbol{R}'_{l,\cdot}$ has to be multiplied by -1 in order to obtain the correct factor. In this case, the l-th row of $\boldsymbol{X}'$ also has to be multiplied by -1. `LINPACK` is not officially maintained anymore, so one probably has to cope with own bug fixes. It is of course easy to wrap `dchex` in order to remove this problem, which is what our implementation does.

References

[1] J. Dongarra, C. Moler, J. Bunch, and G. Stewart. *LINPACK User's Guide.* Society for Industrial and Applied Mathematics, 1979.

[2] P. Gill, G. Golub, W. Murray, and M. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 126(28):505–535, 1974.

[3] D. Goldfarb. Factorized variable metric methods for unconstrained optimization. *Mathematics of Computation*, 30:796–811, 1976.

[4] R. Horn and C. Johnson. *Matrix Analysis.* Cambridge University Press, 1st edition, 1985.