A modified Schur complement method for handling dense columns in interior point methods for linear programming*

Knud D. Andersen†
Dash Associates Ltd
Quinton Lodge, Binswood Avenue, Leamington Spa, Warwickshire, CV32 5TH, UK
E-mail: kda@beta.dou.dk

October 7, 1995

Abstract

The main computational work in interior-point methods for linear programming (LP) is to solve a least-squares problem. The normal equations are often used, but if the LP constraint matrix contains a nearly dense column the normal equations matrix will be nearly dense. Assuming that the non-dense part of the constraint matrix is of full rank, the Schur complement can be used to handle dense columns. In this paper we propose a modified Schur complement method that relaxes this assumption. Encouraging numerical results are presented.

Key words: Linear programming, interior-point methods, Cholesky factorization.

1 Introduction

The most time-consuming part in interior-point methods is to find a solution of the following linear system:

$$\begin{bmatrix} \boldsymbol{D}^{-1} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{0} \end{bmatrix}, \tag{1}$$

where $\boldsymbol{A} \in \Re^{m \times n}$, $\boldsymbol{x} \in \Re^n, \boldsymbol{b} \in \Re^n$ and $\boldsymbol{D} \in \Re^{n \times n}$ is a positive diagonal matrix. (1) can be reduced to a system of normal equations:

$$\boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T\boldsymbol{x} = \boldsymbol{A}\boldsymbol{D}\boldsymbol{b}. \tag{2}$$

We will not discuss methods for solving (1), but only direct methods for (2). We will assume that $\boldsymbol{A}$ has full row rank, so that $\boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T$ is a positive definite symmetric matrix. Alternatives for solving this system are the $\boldsymbol{Q}\boldsymbol{R}$ factorization of $\boldsymbol{D}^{\frac{1}{2}}\boldsymbol{A}^T$, an $\boldsymbol{L}\boldsymbol{U}$ factorization of $\boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T$ using row and column interchanges, or a Cholesky factorization of $\boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T$. The latter is equal to the $\boldsymbol{L}\boldsymbol{U}$ factorization choosing the pivots on the diagonal. The most successful in practice is the Cholesky factorization $\boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T = \boldsymbol{L}\boldsymbol{L}^T$, see e.g. [14]. The success of the Cholesky factorization in interior-point methods is due to its good numerical properties and the fact that it can be implemented very efficiently. If $\boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T$ is sparse, the rows of $\boldsymbol{A}$ can usually be reordered once such that the Cholesky factorization of $\boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T$ is sparse. However a drawback of (2) and the normal equations approach is that if one or more columns of $\boldsymbol{A}$ is nearly dense, $\boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T$ and $\boldsymbol{L}$ will be very dense. Clearly such columns have to be handled in a special way. Several methods have been used to do this:

1. (Schur complement) The Schur complement is used to eliminate dense columns from $\boldsymbol{A}$, such that the remaining part of (2) is sparse [13, 6].

*Technical report, Odense University, Submitted for possible publication in ACM Transactions on Mathematical Software.

†This work was supported by a Danish SNF Research Studentship. Some of it was carried out while the author was a visiting scholar at the Computer Science Department of Stanford University.

2. (Preconditioned conjugate gradient) A PCG method where the preconditioner is a Cholesky factorization of (2), without the dense columns [8, 1].

3. (Splitting) "Stretch" the dense columns into two or more pieces [16].

According to [14], splitting is the most efficient way of handling dense columns in the normal equations system. The problem with the Schur complement has been lack of numerical stability [11, 12], which is very important in interior-point methods. In Section 2 we describe a modified Schur complement, and Section 3 gives implementations details. Finally we present results in Section 4.

2 Theory

In the following we assume that the Cholesky factorization is well known. Let $\boldsymbol{A} = \left[\begin{array}{cc} \widehat{\boldsymbol{A}} & \boldsymbol{E} \end{array} \right]$, where $\boldsymbol{E}$ is a subset of dense columns from $\boldsymbol{A}$. We assume that $\widehat{\boldsymbol{A}}$ has full row rank, and that there are few columns in $\boldsymbol{E}$. We want to solve

$$\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{x} = \widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T\boldsymbol{x} + \boldsymbol{E}\boldsymbol{E}^T\boldsymbol{x} = \boldsymbol{b}. \tag{3}$$

Note that without loss of generality the diagonal matrix $\boldsymbol{D}$ can be omitted, because it is only a scaling of the columns in $\boldsymbol{A}$. (3) is rewritten as

$$\left[\begin{array}{cc} \widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T & \boldsymbol{E} \\ \boldsymbol{E}^T & -\boldsymbol{I} \end{array} \right] \left[\begin{array}{c} \boldsymbol{x} \\ \boldsymbol{r} \end{array} \right] = \left[\begin{array}{c} \boldsymbol{b} \\ \boldsymbol{0} \end{array} \right]. \tag{4}$$

By assumption, $\widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T$ has full rank, so the Cholesky factorization $\boldsymbol{L}\boldsymbol{L}^T = \widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T$ exists, where $\boldsymbol{L}$ is lower triangular. (4) is replaced by the system

$$\left[\begin{array}{cc} \boldsymbol{L}\boldsymbol{L}^T & \boldsymbol{E} \\ \boldsymbol{E}^T & -\boldsymbol{I} \end{array} \right] \left[\begin{array}{c} \boldsymbol{x} \\ \boldsymbol{r} \end{array} \right] = \left[\begin{array}{c} \boldsymbol{b} \\ \boldsymbol{0} \end{array} \right]. \tag{5}$$

If $\boldsymbol{L}\boldsymbol{V} = \boldsymbol{E}$ and $\boldsymbol{L}\boldsymbol{p} = \boldsymbol{b}$ we obtain

$$\begin{aligned} \boldsymbol{L}^T\boldsymbol{x} + \boldsymbol{V}\boldsymbol{r} &= \boldsymbol{p} && (6) \\ \boldsymbol{V}^T\boldsymbol{L}^T\boldsymbol{x} - \boldsymbol{r} &= \boldsymbol{0} && (7) \end{aligned}$$

This way of finding a solution to (3) is known as a Schur complement method (or update); see e.g. [6], [10, 230-231]. The method is formally stated in Figure 1.

$$\begin{aligned} \boldsymbol{L}^T\boldsymbol{L} &= \widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T && (8) \\ \boldsymbol{L}\boldsymbol{V} &= \boldsymbol{E} && (9) \\ \boldsymbol{L}\boldsymbol{p} &= \boldsymbol{b} && (10) \\ (\boldsymbol{V}^T\boldsymbol{V} + \boldsymbol{I})\boldsymbol{r} &= \boldsymbol{V}^T\boldsymbol{p} && (11) \\ \boldsymbol{L}^T\boldsymbol{x} &= \boldsymbol{p} - \boldsymbol{V}\boldsymbol{r} && (12) \end{aligned}$$

Figure 1: The Schur complement method

The vector $\boldsymbol{r}$ can be found cheaply from (11), because of the low dimension of $(\boldsymbol{V}^T\boldsymbol{V} + \boldsymbol{I})$. Cholesky factorization of $(\boldsymbol{V}^T\boldsymbol{V} + \boldsymbol{I})$ is one possibility. Since (11) is equivalent to the least-squares problem

$$\min \left\| \left[\begin{array}{c} \boldsymbol{V} \\ \boldsymbol{I} \end{array} \right] \boldsymbol{r} - \left[\begin{array}{c} \boldsymbol{p} \\ \boldsymbol{0} \end{array} \right] \right\|_2,$$

an alternative is to use a $\boldsymbol{QR}$ factorization of the matrix $\left[\begin{array}{cc} \boldsymbol{V}^T & \boldsymbol{I} \end{array}\right]^T$.

For numerical accuracy and convenience we have tried a third approach, based on an $\boldsymbol{LU}$ factorization of $\boldsymbol{V}^T\boldsymbol{V}+\boldsymbol{I}$ with row and column interchanges for stability. (We use a similar factorization later.) To improve the accuracy of $\boldsymbol{r}$ and $\boldsymbol{L}^T\boldsymbol{x}$, we used iterative refinement [9, pp. 253–254] on (11)–(12) and also on the full system (3). This approach was tested on LP problems with dense columns and proved to work well in most cases. However, it failed as $\widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T$ approached a singular matrix. The other two factorizations would eventually fail for the same reason.

In [11], $\boldsymbol{L}$ is replaced by the Cholesky factor of a modified system,

$$\boldsymbol{L}\boldsymbol{L}^T = \widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T + \rho\boldsymbol{I},$$

where ρ is a positive scalar that ensures the Cholesky factor is well defined. Iterative refinement is then used to compute each column of $\boldsymbol{V}$. This is computationally expensive, and again it does not provide a stable method for solving (3).

Instead of introducing an arbitrary perturbation to the problem, we propose to modify the linear system that is solved. The modified system has the same solution as the original problem but is better conditioned. We know that we can make $\widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T$ of full rank if we add something to the diagonal. Using this observation we rewrite (3) as

$$\left[\begin{array}{ccc} (\widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T + \boldsymbol{F}\boldsymbol{F}^T) & \boldsymbol{E} & \boldsymbol{F} \\ \boldsymbol{E}^T & -\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{F}^T & \boldsymbol{0} & \boldsymbol{I} \end{array}\right] \left[\begin{array}{c} \boldsymbol{x} \\ \boldsymbol{r} \\ \boldsymbol{s} \end{array}\right] = \left[\begin{array}{c} \boldsymbol{b} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{array}\right], \tag{13}$$

where $\boldsymbol{F} \in R^{m\times k}$ is a matrix with only one element different from zero in every column, so that the introduction of $\boldsymbol{F}$ only changes the diagonal of $\widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T$. Therefore the revised matrix has a Cholesky factor with the same sparsity structure as the old. Our modified Schur complement method can now be outlined as in Figure 2.

$$\begin{aligned}
\boldsymbol{L}\boldsymbol{L}^T &= \widehat{\boldsymbol{A}}\widehat{\boldsymbol{A}}^T + \boldsymbol{F}\boldsymbol{F}^T && (14)\\
\boldsymbol{L}\boldsymbol{V} &= \boldsymbol{E} && (15)\\
\boldsymbol{L}\boldsymbol{W} &= \boldsymbol{F} && (16)\\
\boldsymbol{L}\boldsymbol{p} &= \boldsymbol{b} && (17)\\
\boldsymbol{C} &= \left[\begin{array}{c} \boldsymbol{V}^T \\ \boldsymbol{W}^T \end{array}\right] \left[\begin{array}{cc} \boldsymbol{V} & \boldsymbol{W} \end{array}\right] + \left[\begin{array}{cc} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{I} \end{array}\right] && (18)\\
\boldsymbol{C}\left[\begin{array}{c} \boldsymbol{r} \\ \boldsymbol{s} \end{array}\right] &= \left[\begin{array}{c} \boldsymbol{V}^T\boldsymbol{p} \\ \boldsymbol{W}^T\boldsymbol{p} \end{array}\right] && (19)\\
\boldsymbol{L}^T\boldsymbol{x} &= \boldsymbol{p} - \boldsymbol{V}\boldsymbol{r} - \boldsymbol{W}\boldsymbol{s} && (20)
\end{aligned}$$

Figure 2: The modified Schur complement method

In the procedure for calculating the Cholesky factorization (14) we check for rank deficiency by monitoring the size of the diagonal elements. If a diagonal element is small we add something to it and introduce a new column in $\boldsymbol{F}$.

It has been pointed out to us of J. Gondizio that Stewart [15] describes how to handle null pivots exact due to a priory fixed pivot order in Gaussian Elimination on a square matrix of full rank. One reason to such a fixed pivot order is sparsity considerations. The method described here is related.

3 Implementation details

For general implementation details of a sparse Cholesky factorization, see [4]. We will only give implementation details related to the modified Schur complement method in Figure 2..

Normally one would assume that $\boldsymbol{V}$ is fully dense. This is true if the columns in $\boldsymbol{E}$ are fully dense. However, as the problem size increases and columns with more than say 40 nonzeros are dropped, then the columns in $\boldsymbol{E}$ may be sparse, but dense in comparison with the columns in $\widehat{\boldsymbol{A}}$. Note that the number of nonzeros per column in a LP model is typically in the range 3–10. We have therefore chosen to treat $\boldsymbol{V}$ as a sparse matrix. $\boldsymbol{V}$'s symbolic structure is calculated only once, as for the Cholesky factor $\boldsymbol{L}$. Assuming $\boldsymbol{L}$ is known, the nonzero structure of the i'th column of $\boldsymbol{V}$ can be found by solving $\boldsymbol{L}\boldsymbol{V}_i = \boldsymbol{E}_i$ symbolically. This clearly gives a worst-case estimate of nonzeros in $\boldsymbol{V}_i$.

The matrix $\boldsymbol{W}$ is treated the same as $\boldsymbol{V}$.

To solve system (19) we find an $\boldsymbol{LU}$ factorization of the matrix $\boldsymbol{C}$ using row and column interchanges for stability. A $\boldsymbol{QR}$ factorization of $\boldsymbol{C}$ can also be used, but note that the linear system (19) cannot be formulated as a least-squares problem unless $\boldsymbol{W}$ is empty. In practice it is important for satisfactory numerical accuracy to use iterative refinement on (19)–(20).

The number of columns in $\boldsymbol{F}$ is restricted to at most the number of columns in $\boldsymbol{E}$. A column is added permanently to $\boldsymbol{F}$ if $l_{ii} < tol_{add}$, where l_{ii} is the i'th diagonal element of $\boldsymbol{L}$. This is found to work well in practice.

Another important issue is to choose which columns should be in $\boldsymbol{E}$. We keep the number of columns in $\boldsymbol{E}$ relatively small to avoid possible numerical instability, and because it is inefficient to drop many columns. Let n be the number of columns in $\boldsymbol{A}$ and define $\beta = |\boldsymbol{A}|/n$, where $|\boldsymbol{A}|$ is the number of nonzeros in $\boldsymbol{A}$. Let $\boldsymbol{A}_j$ be the j'th column in $\boldsymbol{A}$, and define $n_j = |\boldsymbol{A}_j|$. $\boldsymbol{E}$ is obtained in the following way:

$$
\begin{aligned}
&\boldsymbol{E}^1 = \{\boldsymbol{A}_j \mid n_j > 3\beta, \quad n_j \geq \gamma, \quad i = 1, \ldots, n\} \\
&\boldsymbol{E}^2 = \{\boldsymbol{A}_j \mid n_j > 10\beta, \quad n_j \geq \gamma, \quad i = 1, \ldots, n\} \\
&\text{if (columns in } \boldsymbol{E}^1) \leq \eta \\
&\quad \boldsymbol{E} = \boldsymbol{E}^1 \\
&\text{else} \\
&\quad \boldsymbol{E} = \boldsymbol{E}^2
\end{aligned}
$$

where γ ensures that every column in $\boldsymbol{E}$ has a certain amount of nonzeros, and η ensures that the number of columns is not too large. The result is that only relatively dense columns are dropped, and only if $\boldsymbol{A}$ is sparse.

The parameters we have used are $\gamma = 40$, $\eta = 10$ and $tol_{add} = 10^{-10}$.

4 Results

The ideas presented in Section 2 have been implemented, and are part the APOS system which consist of two codes (which are joint work with E.D. Andersen). The codes are OPTMZE, a primal-dual interior-point code for linear programming (This code is part of the XPRESS modelling system from Dash Associates Ltd), and GOPT (Global Optimizer), a Newton method for minimizing the sum of L1 or L2 norms subject to linear constraints [5]. Both codes use the same Cholesky factorization, to be described in [4]. The codes are written in ANSI C and have been ported to several different computers including PC's, HP 9000/715 workstation, SUN SPARCstation 10, different SGI workstations, and a CONVEX C3240 vector computer.

All testing presented here was done on a CONVEX C3240 at Odense University. The method was tested on all problems with dense columns that we had access to. This includes some problems from the netlib test set [7] and some from another source. The method was also tested on two minimum sum of norms (MSN) problem using GOPT. These are problems "ssu" and "cl399".

The results are shown in Tables 1 and 2. $|\boldsymbol{A}\boldsymbol{A}^T|/2$ is the number of nonzeros in $\boldsymbol{A}\boldsymbol{A}^T$ on and below the diagonal, $|\boldsymbol{L}|$ is the number of nonzeros in $\boldsymbol{L}$, $|\boldsymbol{V}|$ is the number of nonzeros in $\boldsymbol{V}$ and $\boldsymbol{W}$, "$\boldsymbol{V}$ col" is the number of columns in $\boldsymbol{V}$ including columns in $\boldsymbol{W}$, $|\boldsymbol{W}|$ is the number of columns in $\boldsymbol{F}$, m is the number of rows in $\boldsymbol{A}$, NSU stands for No Schur Update and WSU for With Schur Update, CPU is total time in cpu seconds including cross-over [2] to a basic feasible solution and presolving [3], and ITER is the total number of iterations.

We see in Table 1 that $\boldsymbol{V}$ is sparse. For p2, the upper bound on the number of nonzeros in $\boldsymbol{V}$ is $15763 \times 51 = 803913$ but there are only 29646 nonzeros, a density of 4%. Storing $\boldsymbol{V}$ sparse clearly gives

Name	$\|AA^T\|/2$		$\|L\|$			$\|V\|$	V col	W col	m
	NSU	WSU	NSU	WSU	NSU/WSU	WSU	WSU	WSU	
fit1d	291	291	296	296	1	0	0	0	24
fit1p	196878	627	196878	627	314	8215	24	0	627
fit2d	321	321	324	324	1	0	0	0	25
fit2p	4501500	3000	4501500	3000	1500	36784	25	0	3000
israel	10963	4273	11224	4442	2.5	435	3	0	163
pilot	59844	56641	197346	186783	1.1	4729	12	3	1362
p1	492797	48077	1112409	199145	5.6	5916	6	3	4901
p2	232686	65826	1339156	163840	8.2	29646	51	1	15763
p3	2381	1823	3208	2615	1.2	666	12	3	203
seba(max)	51165	949	53492	1035	51.7	3762	14	0	446
ssu (MSN)	405450	18188	405450	45732	8.9	900	2	1	900
cl399 (MSN)	3103949	3024548	57678407	30001436	1.9	12924	2	1	319466

Table 1: Results

Name	CPU			ITER	
	NSU	WSU	NSU/WSU	NSU	WSU
fit1d	10.7	11.5	0.93	21	21
fit1p	115.7	12.2	9.48	16	16
fit2d	104.0	111.1	0.94	22	22
fit2p	11770.2	83.2	141.47	20	20
israel	7.7	6.0	1.28	24	24
pilot	283.5	317.5	0.89	43	43
p1	2720.3	438.4	6.20	43	48
p2	2056.7	944.8	2.18	29	28
p3	3.9	4.3	0.91	16	16
seba(max)	33.5	10.1	3.31	21	21
ssu (MSN)	298.7	37.6	7.9	17	23
cl399 (MSN)	100084	36003	2.8	40	35
total	117491	37979	3.1		

Table 2: Results

a large reduction in memory. If we look at $\boldsymbol{L}$ and ignore the storage for $\boldsymbol{V}$ (it is actually very small in comparison with the nonzeros in $\boldsymbol{L}$), the reduction in nonzeros is between 2 and 10 for most of the problems where dense columns occur. We see that the reduction in CPU time (see Table 2) for the same problems is in the range 1.2 to 6. Clearly for these problems it is beneficial to handle the dense columns separately. Only for two problems did the CPU time increase (pilot and p3), which is consistent with the fact that the reduction in nonzeros in $\boldsymbol{L}$ is low. In all cases standard accuracy is achieved, and only on p1 and ssu did it cost more iterations to get the same accuracy; see Table 2. We further observe in Table 1 that the number of columns in $\boldsymbol{W}$ is very low. The cl399 problem is interesting because the column that is dropped has a density of only $(401/319466) * 100 = 0.13\%$, but it is dense in comparison with the other columns in the problem. The reduction in $\boldsymbol{L}$ is somewhat small (below a factor of 2). However, a reduction in non-zeros from 57 million to 30 million would make a critical difference on many computers.

The method was also used on fit1d and fit2d because they are dual problems of fit1p and fit2p. We see that the time for fit1d and fit2d is nearly the same as for fit1p and fit2p using the Schur complement. This means that for some LP problems the dense columns can be avoided by finding a solution to the dual problem. However, this cannot be done in general for nonlinear problems such as the two MSN examples.

5 Conclusion

In this paper we propose a modified Schur complement approach for handling dense columns when a least-squares problem is solved using the normal equations. As noted by other authors, this approach can be used to handle a few dense columns in large sparse LP problems efficiently. However, our modified Schur complement approach handles the numerical rank or near rank deficiency better than the original approach at a minor cost. This conclusion is confirmed by our (limited) numerical results.

Acknowledgment

I thank Erling D. Andersen for reading earlier drafts of this paper and the Associate Editor Michael Saunders for doing much work on the paper.

References

[1] I. Adler, N. Karmarkar, M.G.C. Resende, and G. Veiga. An implementation of Karmarkar's algorithm for linear programming. *Math. Programming*, 44:297–235, 1989.

[2] E.D. Andersen. Finding all linearly dependent rows in large-scale linear programming. Technical report, 1994.

[3] E.D. Andersen and K.D. Andersen. Presolving in linear programming. Preprint 35, Dept. of Math. and Computer Sci., Odense University, 1993.

[4] E.D. Andersen, K.D. Andersen, and X. Xu. Improved Cholesky decompositions for work-stations and vector computers. Technical report, 1994. In preparation.

[5] K.D. Andersen. An efficient Newton barrier method for minimizing a sum of Euclidean norms. Preprint 42, Dept. of Math. and Computer Sci., Odense University, 1993. To appear in SIAM J. on Optim.

[6] I.C. Choi, C.L. Monma, and D.F. Shanno. Further development of a primal-dual interior point method. *ORSA J. on Comput.*, (2):304–311, 1990.

[7] D.M. Gay. Electronic mail distribution of linear programming test problems. *COAL Newsletter*, 13:10–12, 1985.

[8] P.E. Gill, W. Murray, M.A. Saunders, J.A. Tomlin, and M.H. Wright. On the projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Math. Programming*, 36:183–209, 1986.

[9] G.H. Golub and C.F. Van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore and London, 2nd edition, 1989.

[10] M.T. Heath. Some extensions of an algorithm for sparse linear least squares problems. *SIAM J. Sci. Statist. Comput.*, 3(2):223–237, 1982.

[11] I.J. Lustig, R.E. Marsten, and D.F. Shanno. Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra Appl.*, 20:191–222, 1991.

[12] I.J. Lustig, R.E. Marsten, and D.F. Shanno. On implementing Mehrotra's predictor-corrector interior-point method for linear programming. *SIAM J. on Optim.*, 2(3):435–449, 1992.

[13] A. Marxen. Primal barrier methods for linear programming. Report SOL 89-6, Dept. of Operations Research, Stanford University, Stanford, CA, 1989.

[14] M.A. Saunders. Major Cholesky would feel proud. *ORSA J. on Comput.*, 6(1):94–105, 1994.

[15] G. Stewart. Modifying Pivot Elements in Gaussian Elimination. *Math. Comp.*, 28(126):537–642, 1974.

[16] R.J. Vanderbei. Splitting dense columns in sparse linear systems. *Linear Algebra Appl.*, 152:107–117, 1991.