Splitting Dense Columns in Sparse Linear Systems

Robert J. Vanderbei
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

Submitted by David M. Gay

ABSTRACT

We consider systems of equations of the form

$$AA^T x = b,$$

where A is a sparse matrix having a small number of columns which are much denser than the other columns. These dense columns in A cause AA^T to be very (or even completely) dense, which greatly limits the effectiveness of sparse-matrix techniques for directly solving the above system of equations. In the literature on interior-point methods for linear programming, the usual technique for dealing with this problem is to split A into a sparse part S and a dense part D,

$$A = [\,S \quad D\,],$$

and to solve systems involving AA^T in terms of the solution of systems involving SS^T using either conjugate-gradient techniques or the Sherman-Morrison-Woodbury formula. This approach has the difficulty that SS^T is often rank-deficient even when AA^T has full rank. In this paper we propose an alternative method which avoids the rank-deficiency problem and at the same time allows for the effective use of sparse-matrix techniques. The resulting algorithm is both efficient and robust.

1. INTRODUCTION

Let A be a sparse matrix having full row rank. We suppose that some of the columns of A are much denser than the other columns and that these dense columns have been identified. Then we can write A as a partitioned

matrix:

$$A = [S \quad D],$$

where S represents the sparse part of A, and D the dense part. The matrix AA^T is equal to $SS^T + DD^T$, and even though we expect SS^T to be quite sparse, DD^T will be very (or even completely) dense. Hence, AA^T is also very dense, and so it is important to look for a method for solving the system of equations

$$AA^T x = b \tag{1.1}$$

without using AA^T explicitly. Two techniques are commonly used for this purpose. The first technique is to use $(SS^T)^{-1}$ as a preconditioner for the conjugate-gradient method (see e.g. [1–4]). The second technique stays within the framework of direct methods. It is based on the Sherman-Morrison-Woodbury formula:

$$(AA^T)^{-1} = (SS^T)^{-1} - (SS^T)^{-1} D\left[I + D^T (SS^T)^{-1} D\right]^{-1} D^T (SS^T)^{-1}$$

(see e.g. [5–7]). These approaches both have the problem that they require SS^T to be of full rank, but in practice SS^T is often rank-deficient. There have been several suggestions to overcome this problem (see e.g. [8, 7, 9, 10]) but so far all attempts have resulted in algorithms that are quite sensitive to numerical error and require careful tuning of parameters.

This paper presents an alternative method for solving (1.1) which is efficient and robust. The method is motivated by the idea of splitting dense columns in the constraint matrix of linear programming problems. In Section 2, we briefly discuss the Sherman-Morrison-Woodbury formula. Then in Section 3, we describe how the method works in the case where the matrix has a single dense column. In Section 4, we study the general case. Then in Section 5, we discuss the application of these techniques to interior-point methods for linear programming. Finally, in Section 6 we extend the method to the case of linear systems that are not necessarily symmetric.

2. THE SHERMAN-MORRISON-WOODBURY FORMULA

Consider the following system of equations:

$$\begin{bmatrix} SS^T & D \\ D^T & -I \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{2.1}$$

which is written out as

$$SS^T x + Dy = b, \tag{2.2}$$

$$D^T x - y = 0. \tag{2.3}$$

If we solve (2.3) for y as a function of x and then substitute this into (2.2), we see that

$$x = (SS^T + DD^T)^{-1} b = (AA^T)^{-1} b.$$

Now, suppose that SS^T is invertible, so that we can solve (2.2) for x as a function of y:

$$x = (SS^T)^{-1}(b - Dy). \tag{2.4}$$

Then we can substitute this into (2.3) and obtain

$$y = \left[I + D^T(SS^T)^{-1} D\right]^{-1} D^T(SS^T)^{-1} b.$$

Finally, substituting this expression for y into (2.4), we get the Sherman-Morrison-Woodbury formula for $(SS^T + DD^T)^{-1}$:

$$x = \left\{(SS^T)^{-1} - (SS^T)^{-1} D\left[I + D^T(SS^T)^{-1} D\right]^{-1} D^T(SS^T)^{-1}\right\} b. \tag{2.5}$$

Using (2.1) as a starting point for deriving the Sherman-Morrison-Woodbury formula motivates us to ask: why not solve (2.1) with a general linear system solver and then just keep the x part of the answer? One would think that this should do even better than the Sherman-Morrison-Woodbury formula, since it does not predetermine the order in which the equations are solved. However, it suffers from the drawback that the matrix in (2.1) is not positive definite (even though it is symmetric). Hence, for direct factorization methods, pivoting rules depending on tolerances become quite important. The splitting technique described in the next two sections of this paper is similar to this approach except that it always involves symmetric positive definite matrices.

3. SPLITTING A SINGLE DENSE COLUMN

Suppose that D consists of a single dense column d:

$$A = [S \quad d].$$

We split d into a k-column matrix Δ by distributing to each column a few nonzeros from d. The matrix Δ is related to d by the following formula:

$$\Delta e = d,$$

where e is a k-vector of ones. Of course, to be efficient we want the columns of Δ to be sparse, but this is not necessary for the algebra. Consider the system of equations

$$CC^T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{3.1}$$

where

$$C = \begin{bmatrix} S & \sqrt{k}\,\Delta \\ 0 & L \end{bmatrix},$$

and

$$L = \begin{bmatrix} 1 & -1 & & & & & \\ & 1 & -1 & & & & \\ & & 1 & -1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & -1 \end{bmatrix}.$$

The $(k-1)\times k$ matrix L is called a *linking matrix*.

THEOREM 1. *C has full row rank if and only if A has full row rank. Assuming that C has full row rank, let $[x^T \; y^T]^T$ be the solution to* (3.1). *Then*

$$x = (AA^T)^{-1} b.$$

Proof. Let m and n denote the numbers of rows and columns of A, respectively. Then C has $m+k-1$ rows and $n+k-1$ columns. Since

performing column operations does not affect the rank of a matrix, we can replace the nth column of C (i.e. the first in the split group of columns) with the sum of columns $n, n+1, \ldots, n+k-1$ to get the following matrix, which has the same rank as C:

$$\begin{bmatrix} S & d & \Delta_2 & \cdots & \Delta_k \\ 0 & 0 & L_2 & \cdots & L_k \end{bmatrix} = \begin{bmatrix} A & \Delta_2 & \cdots & \Delta_k \\ 0 & L_2 & \cdots & L_k \end{bmatrix},$$

where Δ_j denotes the jth column of Δ and L_j denotes the jth column of L. We now see that this matrix has full row rank if and only if A does, since the submatrix

$$\begin{bmatrix} L_2 & \cdots & L_k \end{bmatrix}$$

is an invertible $(k-1)\times(k-1)$ matrix.

Now assume that C has full row rank. Then writing out (3.1), we see that

$$(SS^T + k\Delta\Delta^T)x + \sqrt{k}\,\Delta L^T y = b, \tag{3.2}$$

$$\sqrt{k}\,L\Delta^T x + LL^T y = 0. \tag{3.3}$$

Solving (3.3) for y, we get

$$y = -\sqrt{k}\,(LL^T)^{-1} L\Delta^T x,$$

and substituting this into (3.2), we get the following expression for x:

$$x = \left\{SS^T + k\Delta\left[I - L^T(LL^T)^{-1}L\right]\Delta^T\right\}^{-1} b. \tag{3.4}$$

If we can show that

$$I - L^T(LL^T)^{-1}L = \frac{1}{k}ee^T, \tag{3.5}$$

we will be done, since then (3.4) becomes

$$x = (SS^T + dd^T)^{-1} b = (AA^T)^{-1} b.$$

To prove (3.5), we first note that the left-hand side is the projection onto the null space of L. But from the definition of L, we see that its null space is the one-dimensional space spanned by e. The projection onto this one-dimensional space is easily seen to be given by ee^T/k. ■

Theorem 1 says that instead of solving the system (1.1), one could instead solve the split system (3.1) for x. Even though the system (3.1) is larger, it has a better sparsity structure and can often be solved much faster than the original system.

4. SPLITTING SEVERAL DENSE COLUMNS

Now we return to the general case where A can have several dense columns:

$$A = [S \quad D].$$

A splitting (and scaling) of D is a matrix Δ having the same number of rows but more columns and which satisfies

$$\Delta E = D, \tag{4.1}$$

where

$$E = \begin{bmatrix} \frac{1}{\sqrt{k_1}} e_1 & & & \\ & \frac{1}{\sqrt{k_2}} e_2 & & \\ & & \ddots & \\ & & & \frac{1}{\sqrt{k_n}} e_n \end{bmatrix}, \tag{4.2}$$

where each e_i is a k_i-vector of all ones. Hence, we see that we have expanded the ith column of D into k_i new sparse columns. For each i, let

L_i be a $(k_i - 1) \times k_i$ linking matrix, and let

$$L = \begin{bmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_n \end{bmatrix}. \tag{4.3}$$

Now, consider the system of equations

$$CC^T \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{4.4}$$

where

$$C = \begin{bmatrix} S & \Delta \\ 0 & L \end{bmatrix}.$$

THEOREM 2. *C has full row rank if and only if A has full row rank. Assuming that C has full row rank, let $[x^T \ y^T]^T$ be the solution to* (4.4). *Then*

$$x = (AA^T)^{-1} b.$$

Proof. The proof that C has full row rank if and only if A does proceeds in the same way as the proof of the same statement in Theorem 1, so we omit it here.

Again performing block matrix algebra, we see that

$$x = \left\{ SS^T + \Delta \left[I - L^T (LL^T)^{-1} L \right] \Delta^T \right\}^{-1} b. \tag{4.5}$$

If we can show that

$$I - L^T (LL^T)^{-1} L = EE^T, \tag{4.6}$$

we will be done, since then (4.5) becomes

$$x = (SS^T + DD^T)^{-1} b = (AA^T)^{-1} b.$$

To prove (4.6), we again note that the left-hand side is the projection onto

the null space of L. But from the definition of L, we see that its null space is the n-dimensional space spanned by the following n orthonormal vectors:

$$\frac{1}{\sqrt{k_1}}\begin{bmatrix} e_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \frac{1}{\sqrt{k_2}}\begin{bmatrix} 0 \\ e_2 \\ \vdots \\ 0 \end{bmatrix}, \ldots, \quad \frac{1}{\sqrt{k_n}}\begin{bmatrix} 0 \\ 0 \\ \vdots \\ e_n \end{bmatrix}.$$

Projection onto this space is easily seen to be given by EE^T. ■

5. APPLICATION TO INTERIOR-POINT METHODS FOR LINEAR PROGRAMMING

Consider the following linear programming problem:

$$\begin{aligned} &\text{minimize} \quad c^T x: \\ &Ax = b, \\ &x \geqslant 0. \end{aligned}$$

Interior-point methods for solving this problem rely on being able to solve efficiently systems of equations of the form

$$AX^2A^T y = d, \tag{5.1}$$

where X is a diagonal matrix. Hence, dense columns in A are an impediment to efficient interior-point algorithms for linear programming based on direct factorization methods.

As before, let us suppose that we have identified the dense columns, so that we can rewrite the linear programming problem in block matrix form:

$$\begin{aligned} &\text{minimize} \quad c_S^T x_S + c_D^T x_D: \\ &\begin{bmatrix} S & D \end{bmatrix}\begin{bmatrix} x_S \\ x_D \end{bmatrix} = b, \\ &x_S \geqslant 0, \qquad x_D \geqslant 0. \end{aligned}$$

One technique for dealing with dense columns (see e.g. [11] or [12]) is to

split each column of D into several columns (thereby creating additional variables) and to force the corresponding variables to be equal by including linking constraints. Algebraically, this derived LP can be written as

$$\text{minimize} \quad c_S^T x_S + c_\Delta^T x_\Delta:$$

$$\begin{bmatrix} S & \Delta \\ 0 & L \end{bmatrix} \begin{bmatrix} x_S \\ x_\Delta \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix},$$

$$x_S \geqslant 0, \qquad x_\Delta \geqslant 0,$$

where Δ is as defined in (4.1) and L is as defined in (4.3). It is easy to see that if c_Δ is chosen properly and if x_Δ is related back to x_D properly, then this new LP is equivalent to the original one.

This technique of splitting the dense columns has been used in an experimental LP code called ALPO (another Linear Programming Optimizer) and has proven to be quite effective (see [13]). For example, the solution time for FIT2P (from the NETLIB suite of test problems—see [14]) has been speeded up by a factor of 120, and SEBA (another NETLIB problem) has been speeded up by a factor of 10.

Splitting dense columns in a linear program motivated the method described in the previous section for linear systems, but the two splitting techniques are not identical. The difference stems from the diagonal matrix that appears between A and A^T in the linear-programming context and also from the fact that the right-hand side of the expanded version of (5.1) does not have a vanishing component as it did in the linear-systems context. There are some advantages in doing the splitting in the context of linear systems as opposed to the way described in this section. First, putting the splitting code in with the numerical code for solving systems of equations isolates the splitting from the LP code. This yields better program structure. For example, if one wants to write out intermediate solutions after every iteration, it would be necessary to reconsolidate the problem if the splitting were done at the LP level, whereas no reconsolidation is required if the splitting is done at the equation-solving level. Also, by pushing the splitting down to the equation-solving level, one gains some efficiency at the LP level because the algorithms at the LP level can act on the original unexpanded linear programming problem.

The effectiveness of these splitting techniques depends on how good the heuristic is which identifies and decides exactly how to split dense columns. In ALPO, the heuristic is of the simplest possible kind. Namely, there is a threshold parameter θ, and any column with more than θ nonzeros is split

into a set of columns, each containing exactly θ nonzeros, except for the last column, which contains the remainder of the nonzeros. The nonzeros are allocated to the new columns in the order in which they appear in the sparse-matrix data structure. That is, the first θ nonzeros go into the first column, the second θ go into the second column, etc. It would be interesting to develop heuristics which are fast but which are more sophisticated than the present one.

6. GENERAL LINEAR SYSTEMS

Suppose we wish to solve

$$Ax = b, \tag{6.1}$$

where A has the form

$$A = B + CD^T$$

with B being sparse and both C and D being dense. Let Γ and Δ be splittings of C and D, respectively. That is,

$$\Gamma E = C,$$
$$\Delta E = D,$$

where E is given by (4.2). Then instead of solving the comparatively dense system (6.1), we solve the following much sparser system

$$\begin{bmatrix} B + \Gamma\Delta^T & \Gamma L^T \\ L\Delta^T & LL^T \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix},$$

where L is given by (4.3). Then mimicking the proof of Theorem 2, we see that the x so obtained is actually a solution to $Ax = b$.

The author would like to thank R. Fourer and D. Gay for interesting discussions which motivated this work.

REFERENCES

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

2 I. Adler, N. K. Karmarkar, M. G. C. Resende, and G. Veiga, An implementation of Karmarkar's algorithm for linear programming, *Math. Programming* 44:297–335 (1989).

3 I. Adler, N. K. Karmarkar, M. G. C. Resende, and G. Veiga, Data structures and programming techniques for the implementation of Karmarkar's algorithm, *ORSA J. Comput.* 1:84–106 (1989).

4 S. Mehrotra, Implementations of Affine Scaling Methods: Approximate Solutions of Systems of Linear Equations Using Preconditioned Conjugate Gradient Methods, Technical Report 89-04, Dept. of Industrial Engineering and Management Science, Northwestern Univ., Evanston, Ill., 1989.

5 I. C. Choi, C. L. Monma, and D. F. Shanno, Further Development of a Primal-Dual Interior Point Method, Technical Report RRR 60-88, RUTCOR, Rutgers Univ., 1988.

6 P. E. Gill, W. Murray, and M. A. Saunders, A Single-Phase Dual Barrier Method for Linear Programming, Technical Report SOL 88-10, Systems Optimization Lab. Stanford Univ., Stanford, Calif., 1988.

7 I. J. Lustig, R. E. Marsten, and D. F. Shanno, Computational Experience with a Primal-Dual Interior Point Method for Linear Programming, Technical Report SOR 89-17, Dept. of Civil Engineering and Operations Research, Princeton Univ., Oct. 1989.

8 R. E. Marsten, M. J. Saltzman, D. F. Shanno, G. S. Pierce, and J. F. Ballintijn, Implementation of a Dual Interior Point Algorithm for Linear Programming, Technical Report RRR 44-88, RUTCOR, Rutgers Univ., New Brunswick, N.J., 1988.

9 K. A. McShane, C. L. Monma, and D. F. Shanno, An implementation of a primal-dual interior point method for linear programming, *ORSA J. Comput.* 1:70–83 (1989).

10 I. J. Lustig, R. E. Marsten, and D. F. Shanno, On Implementing Mehrotra's Predictor-Corrector Interior Point Method for Linear Programming, Technical Report SOR 90-03, Dept. of Civil Engineering and Operations Research, Princeton Univ., Apr. 1990.

11 R. J. Vanderbei, ALPO: Another Linear Program Optimizer, Technical Report, AT&T Bell Labs., 1990.

12 I. J. Lustig, J. M. Mulvey, and T. J. Carpenter, Formulating Stochastic Programs for Interior Point Methods, Technical Report SOR 89-16, Dept. of Civil Engineering and Operations Research, Princeton Univ., Sept. 1989.

13 R. J. Vanderbei, A Brief Description of ALPO, Technical Report, AT&T Bell Labs., 1990.

14 D. M. Gay, Electronic mail distribution of linear programming test problems, *Math. Programming Soc. COAL Newslett.*, 1985.

Received 22 August 1990; final manuscript accepted 25 September 1990