SIAM/ACM Preprint Series Macros for Use With LaTeX*

Corey Gray† Tricia Manning‡

Abstract

This is the text of my abstract. It is a brief description of my paper, outlining the purposes and goals I am trying to address.

1 Problem Specification.

In this paper, we consider the solution of the $N \times N$ linear system

$$Ax = b \tag{1.1}$$

where A is large, sparse, symmetric, and positive definite. We consider the direct solution of (1.1) by means of general sparse Gaussian elimination. In such a procedure, we find a permutation matrix P, and compute the decomposition

$$PAP^t = LDL^t$$

where L is unit lower triangular and D is diagonal.

2 Design Considerations.

Several good ordering algorithms (nested dissection and minimum degree) are available for computing P [3], [7]. Since our interest here does not focus directly on the ordering, we assume for convenience that $P = I$, or that A has been preordered to reflect an appropriate choice of P.

Our purpose here is to examine the nonnumerical complexity of the sparse elimination algorithm given in [1]. As was shown there, a general sparse elimination scheme based on the bordering algorithm requires less storage for pointers and row/column indices than more traditional implementations of general sparse elimination. This is accomplished by exploiting the m-tree, a particular spanning tree for the graph of the filled-in matrix.

THEOREM 2.1. *The method was extended to three dimensions. For the standard multigrid coarsening (in which, for a given grid, the next coarser grid has* 1/8 *as many points), anisotropic problems require plane relaxation to obtain a good smoothing factor.*

Our purpose here is to examine the nonnumerical complexity of the sparse elimination algorithm given in [1]. As was shown there, a general sparse elimination scheme based on the bordering algorithm requires less storage for pointers and row/column indices than more traditional implementations of general sparse elimination. This is accomplished by exploiting the m-tree, a particular spanning tree for the graph of the filled-in matrix. Several good ordering algorithms (nested dissection and minimum degree) are available for computing P [3], [7]. Since our interest here does not focus directly on the ordering, we assume for convenience that $P = I$, or that A has been preordered to reflect an appropriate choice of P.

Proof. In this paper we consider two methods. The first method is basically the method considered with two differences: first, we perform plane relaxation by a two-dimensional multigrid method, and second, we use a slightly different choice of interpolation operator, which improves performance for nearly singular problems. In the second method coarsening is done by successively coarsening in each of the three independent variables and then ignoring the intermediate grids; this artifice simplifies coding considerably. ∎

*The full version of the paper can be accessed at `https://arxiv.org/abs/1902.09310`

†Society for Industrial and Applied Mathematics.

‡Society for Industrial and Applied Mathematics.

Our purpose here is to examine the nonnumerical complexity of the sparse elimination algorithm given in [1]. As was shown there, a general sparse elimination scheme based on the bordering algorithm requires less storage for pointers and row/column indices than more traditional implementations of general sparse elimination. This is accomplished by exploiting the m-tree, a particular spanning tree for the graph of the filled-in matrix.

DEFINITION 2.1. We describe the two methods in §1.2. In § 1.3. we discuss some remaining details.

Our purpose here is to examine the nonnumerical complexity of the sparse elimination algorithm given in [1]. As was shown there, a general sparse elimination scheme based on the bordering algorithm requires less storage for pointers and row/column indices than more traditional implementations of general sparse elimination. This is accomplished by exploiting the m-tree, a particular spanning tree for the graph of the filled-in matrix. Several good ordering algorithms (nested dissection and minimum degree) are available for computing P [3], [7]. Since our interest here does not focus directly on the ordering, we assume for convenience that $P = I$, or that A has been preordered to reflect an appropriate choice of P.

Our purpose here is to examine the nonnumerical complexity of the sparse elimination algorithm given in [1]. As was shown there, a general sparse elimination scheme based on the bordering algorithm requires less storage for pointers and row/column indices than more traditional implementations of general sparse elimination.

LEMMA 2.1. *We discuss first the choice for I_{k-1}^k which is a generalization. We assume that G^{k-1} is obtained from G^k by standard coarsening; that is, if G^k is a tensor product grid $G_x^k \times G_y^k \times G_z^k$, $G^{k-1} = G_x^{k-1} \times G_y^{k-1} \times G_z^{k-1}$, where G_x^{k-1} is obtained by deleting every other grid point of G_x^k and similarly for G_y^k and G_z^k.*

To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [4] - [10], [5], [6]. In §1.3., we analyze the complexity of the old and new approaches to the intersection problem for the special case of an $n \times n$ grid ordered by nested dissection. The special structure of this problem allows us to make exact estimates of the complexity. To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [4] - [10], [5], [6].

In §1.2, we review the bordering algorithm, and introduce the sorting and intersection problems that arise in the sparse formulation of the algorithm. In §1.3., we analyze the complexity of the old and new approaches to the intersection problem for the special case of an $n \times n$ grid ordered by nested dissection. The special structure of this problem allows us to make exact estimates of the complexity. To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [4] - [10], [5], [6].

For the old approach, we show that the complexity of the intersection problem is $O(n^3)$, the same as the complexity of the numerical computations. For the new approach, the complexity of the second part is reduced to $O(n^2(\log n)^2)$.

To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [4] - [10], [5], [6]. In §1.3., we analyze the complexity of the old and new approaches to the intersection problem for the special case of an $n \times n$ grid ordered by nested dissection. The special structure of this problem allows us to make exact estimates of the complexity. To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [4] - [10], [5], [6]. This is accomplished by exploiting the m-tree, a particular spanning tree for the graph of the filled-in matrix. To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [2] - [6], [8], [10].

2.1 Robustness. We do not attempt to present an overview here, but rather attempt to focus on those results that are relevant to our particular algorithm. This section assumes prior knowledge of the role of graph theory

Figure 1: This is a figure 1.1.

in sparse Gaussian elimination; surveys of this role are available in [7] and [3]. More general discussions of elimination trees are given in [4] - [6], [10]. Thus, at the kth stage, the bordering algorithm consists of solving the lower triangular system

$$L_{k-1}v = c \tag{2.2}$$

and setting

$$\ell = D_{k-1}^{-1}v, \tag{2.3}$$

$$\delta = \alpha - \ell^t v. \tag{2.4}$$

3 Robustness.

We do not attempt to present an overview here, but rather attempt to focus on those results that are relevant to our particular algorithm.

3.1 Versatility. The special structure of this problem allows us to make exact estimates of the complexity. For the old approach, we show that the complexity of the intersection problem is $O(n^3)$, the same as the complexity of the numerical computations [3], [9]. For the new approach, the complexity of the second part is reduced to $O(n^2(\log n)^2)$.

To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [4] - [10], [5], [6]. In §1.3., we analyze the complexity of the old and new approaches to the intersection problem for the special case of an $n \times n$ grid ordered by nested dissection. The special structure of this problem allows us to make exact estimates of the complexity. To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [4] - [10], [5], [6].

In §1.2, we review the bordering algorithm, and introduce the sorting and intersection problems that arise in the sparse formulation of the algorithm. In §1.3., we analyze the complexity of the old and new approaches to the intersection problem for the special case of an $n \times n$ grid ordered by nested dissection. The special structure of this problem allows us to make exact estimates of the complexity. To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [4] - [10], [5], [6].

For the old approach, we show that the complexity of the intersection problem is $O(n^3)$, the same as the complexity of the numerical computations. For the new approach, the complexity of the second part is reduced to $O(n^2(\log n)^2)$.

To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [4] - [10], [5], [6]. In §1.3., we analyze the complexity of the old and new approaches to the intersection problem for the special case of an $n \times n$ grid ordered by nested dissection. The special structure of this problem allows us to make exact estimates of the complexity. To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [4] - [10], [5], [6]. This is accomplished by exploiting the m-tree, a particular spanning tree for the graph of the filled-in matrix. To our knowledge, the m-tree previously has not been applied in this fashion to the numerical factorization, but it has been used, directly or indirectly, in several optimal order algorithms for computing the fill-in during the symbolic factorization phase [2] - [6], [8], [10].

References

[1] R. E. Bank and R. K. Smith, *General sparse elimination requires no permanent integer storage*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 574–584.

[2] S. C. Eisenstat, M. C. Gursky, M. Schultz, and A. Sherman, *Algorithms and data structures for sparse symmetric gaussian elimination*, SIAM J. Sci. Stat. Comput., 2 (1982), pp. 225–237.

[3] A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.

[4] K. H. Law and S. J. Fenves, *A node addition model for symbolic factorization*, ACM TOMS, 12 (1986), pp. 37–50.

[5] J. W. H. Liu, *A compact row storage scheme for cholesky factors using elimination trees*, ACM TOMS, 12 (1986), pp. 127–148.

[6] ———, *The role of elimination trees in sparse factorization*, Tech. Report CS-87-12,Department of Computer Science, York University, Ontario, Canada, 1987.

[7] D. J. Rose, *A graph theoretic study of the numeric solution of sparse positive definite systems*, in Graph Theory and Computing, Academic Press, New York, 1972.

[8] D. J. Rose, R. E. Tarjan, and G. S. Lueker, *Algorithmic aspects of vertex elimination on graphs*, SIAM J. Comput., 5 (1976), pp. 226–283.

[9] D. J. Rose and G. F. Whitten, *A recursive analysis of disection strategies*, in Sparse Matrix Computations, Academic Press, New York, 1976.

[10] R. Schrieber, *A new implementation of sparse gaussian elimination*, ACM TOMS, 8 (1982), pp. 256–276.