Anonymized SIAM/ACM Preprint Series Macros for Use With LaTeX*

Submission ID XYZ

Abstract

An abstract is a brief summary of the paper's contributions, written for experts. Authors are anonymized for double-blind review.

1 Introduction.

An introduction is a gentler description and summary of the paper than the abstract, written for non-experts. It describes the paper's concepts, contribution, context and significance.

1.1 Problem Specification.
In this paper, we consider the solution of the $N \times N$ linear system

$$(1.1) \qquad Ax = b$$

where A is large, sparse, symmetric, and positive definite. We consider the direct solution of eq. (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; see Thm. 2.1. 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. □

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 section 2. In section 2.1 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, 5, 6, 7, 8, 9, 10]. In section 2.1, 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, 5, 6, 7, 8, 9, 10].

In section 2, we review the bordering algorithm, and introduce the sorting and intersection problems that arise in the sparse formulation of the algorithm. In section 2.1, 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

Figure 1: This is a blank figure.

algorithms for computing the fill-in during the symbolic factorization phase [4, 5, 6, 7, 8, 9, 10].

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, 5, 6, 7, 8, 9, 10]. In section 2.1, 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, 5, 6, 7, 8, 9, 10]. 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, 3, 4, 5, 6, 8, 10].

We show a blank figure in fig. 1.

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; see fig. 1. This section assumes prior knowledge of the role of graph theory in sparse Gaussian elimination;

surveys of this role are available in [7, 3]. More general discussions of elimination trees are given in [4, 5, 6, 10]. Thus, at the kth stage, the bordering algorithm consists of solving the lower triangular system

$$(2.2) \qquad\qquad L_{k-1}v = c$$

and setting

$$(2.3) \qquad\qquad \ell \;=\; D_{k-1}^{-1}v,$$
$$(2.4) \qquad\qquad \delta \;=\; \alpha - \ell^t v.$$

3 Algorithm

We provide some pseudocode in Algorithm 3.1.

ALGORITHM 3.1. (DETERMINISTIC-MPS) maximal Poisson-disk sampling

Require: Rectangular grid $\mathcal{G}$ of whole grid squares
Require: Flag if domain is periodic: `True` or `False`
Ensure: Maximal Poisson-disk sampling of rectangle
    function DETERMINISTIC-MPS($\mathcal{G}$)
        // Initialize Grid $\mathcal{G}$
3:      **for** $g \in \mathcal{G}$ **do**
          g.point $= (u,v)$ uniform random in square
          g.time $= Ae^{-Aw}$, rand w, expovariate in area
6:         g.scooped-square $=$ square polygon g
      end for
      Global pre-pass heuristic
9:      // Find locally-early squares
      for $g \in \mathcal{G}$ and $h \in$ neighbors(g) **do**
         increment #antecedents of g or h (later)
12:     **end for**
      for $g \in \mathcal{G}$ **do**
         EarlySquares.add(g if no antecedents)
15:     **end for**
      // Accept samples and update
      repeat
18:     $g =$ EarlySquares.pop()   ▷ any order
         accept g.point as Poisson-disk sample
         for $h \in$ neighbors(g) **do**
21:         decrement h.antecedents
                 ▷ since g no longer blocks h
         // resample candidates in disk(g.point)
24:        **if** h.point $\in$ disk(g.point) **then**
           h.scooped-square $-=$ disk(g.point)
           if h.scooped-square is empty **then**
27:             h.time $= \infty$
           else
             trim chocks of h.scooped-square
30:            triangulate remaining polygon
             $U \in \{$chocks, triangles$\}$ by area
             h.point $\in U$ uniform by area
33:            h.time $+=$ expovar(
   A(h.scooped-square))

           end if
          for $s \in$ neighbors(h) **do**
36:        **if** h is later than s, but was earlier
   then
             increment h.antecedents
             decrement s.antecedents
39:          **if then**s has no antecedents
             EarlySquares.add(s)
          end if
42:        **end if**
         end for
        end if
45:      **if then**h has no antecedents
         EarlySquares.add(h)
      end if
48:    **end for**
   until EarlySquares == empty
  end function

4 Results.

We do not attempt to present an overview here, but rather attempt to focus on those results that are relevant to our particular algorithm, Algorithm 3.1.

4.1 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, 5, 6, 7, 8, 9, 10]. In section 2.1, 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, 5, 6, 7, 8, 9, 10].

In section 2, we review the bordering algorithm, and introduce the sorting and intersection problems that arise in the sparse formulation of the algorithm. In section 2.1, 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, 5, 6, 7, 8, 9, 10].

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, 5, 6, 7, 8, 9, 10]. In section 2.1, 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.

Acknowledgements

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. Rep. 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, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976.

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