POLLACK PERIODICA
An International Journal for Engineering and Information Sciences
DOI: 10.1556/Pollack.4.2009.3.11
Vol. 4, No. 3, pp. 121–128 (2009)
www.akademiai.com

ASSESSMENTS OF THE IMPLEMENTATION OF THE MINIMUM DEGREE ORDERING ALGORITHMS

[1] Petr PAŘÍK, [2] Jiří PLEŠEK

Institute of Thermomechanics, Academy of Sciences of the Czech Republic, Dolejškova 5
182 00 Praha 8, Czech Republic, e-mail: [1] parik@it.cas.cz, [2] plesek@it.cas.cz

Received 9 February 2009; accepted 12 March 2009

Abstract: The minimum degree ordering is one of the most widely used algorithms to preorder a symmetric sparse matrix prior to numerical factorization. There are number of variants which try to reduce the computational complexity of the original algorithm while maintaining a reasonable ordering quality. An in-house finite element solver is used to test several minimum degree algorithms to find the most suitable configuration for the use in the Finite Element Method. The results obtained and their assessments are presented along with the minimum degree ordering algorithms overview.

Keywords: Minimum degree ordering, Direct methods, Sparse solver, Matrix factorization

1. Introduction

Matrices obtained from the finite element discretization can have very large dimensions, depending on the model complexity. Fortunately, these matrices are symmetric and *sparse*, i.e. only a small percentage of matrix elements are non-zero - often less than a percent [1]. Therefore, it is not necessary to store and manipulate all matrix elements - or approximately one half in the case of symmetric matrices - which would be actually impossible even on today's computers (e.g. a relatively 'small' symmetric matrix of order 10^5 would take about 40 gigabytes of storage space).

Another problem is the *fill-in* - the change of originally zero matrix elements into non-zeros during solution. In the worst-case scenario, it is quite possible to get a full triangular matrix (factor) from a symmetric sparse matrix with only a few non-zero elements by factorization [2]. It is therefore paramount to reorder the matrix rows and

columns - actually to reorder the system of linear equations - in a way the fill-in would be minimized or reasonably reduced so the sparsity of the matrix could be preserved to some extent.

The minimum degree ordering [3] is one of the most widely used reordering algorithms, because it produces matrix factors with relatively low fill-in over a wide range of matrices [4]. *Fig. 1* shows an example of the structure of a matrix and its factor when two different ordering methods were applied.

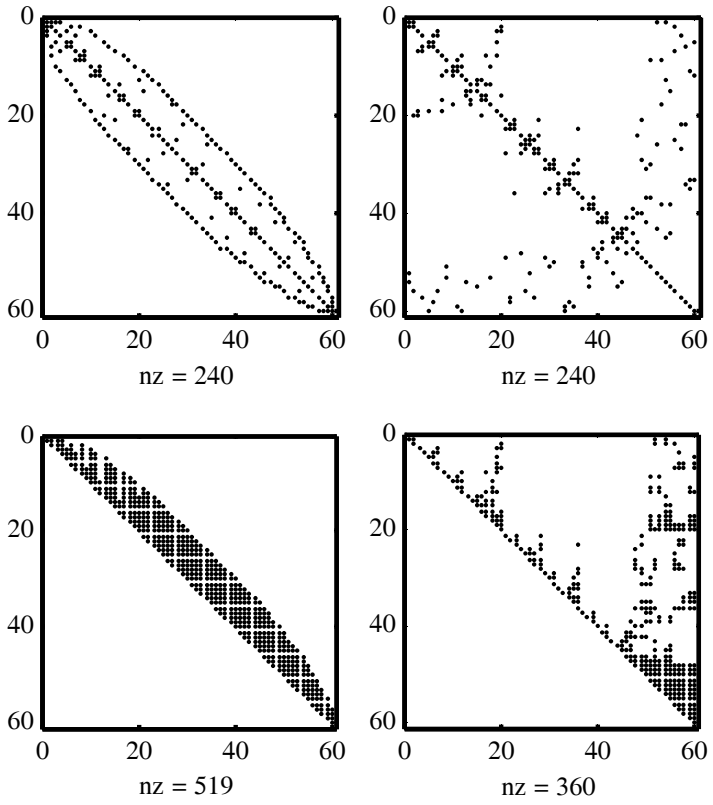

Fig. 1. Example of matrix structure after reordering (top) and after factorization (bottom) Reverse Cuthill-McKee ordering (left) and Minimum Degree ordering (right)

2. Minimum degree ordering

The purpose of the minimum degree ordering is to find such reordering of matrix rows and columns that the numerical factorization on the reordered matrix results in the least fill-in, thus minimizing both storage and time required for the solution [3]. It should be noted that the problem of finding the least fill-in is NP-complete, so the minimum degree ordering actually uses a heuristic algorithm to find the 'almost optimal' ordering.

The original minimum degree algorithm has some shortcomings, so several ways to improve it were devised over time [4], [5], [6]. The most significant improvements are briefly described in this section.

2.1. Original algorithm

The original minimum degree algorithm [3], [4], [5] is based on the *elimination graph* (see *Fig. 2*): vertices represent pivots (i.e. diagonal elements) of the matrix, while edges represent non-zero elements of the matrix. Moreover, each vertex has a *degree* defined as a number of adjacent vertices, i.e. the number of vertex edges.

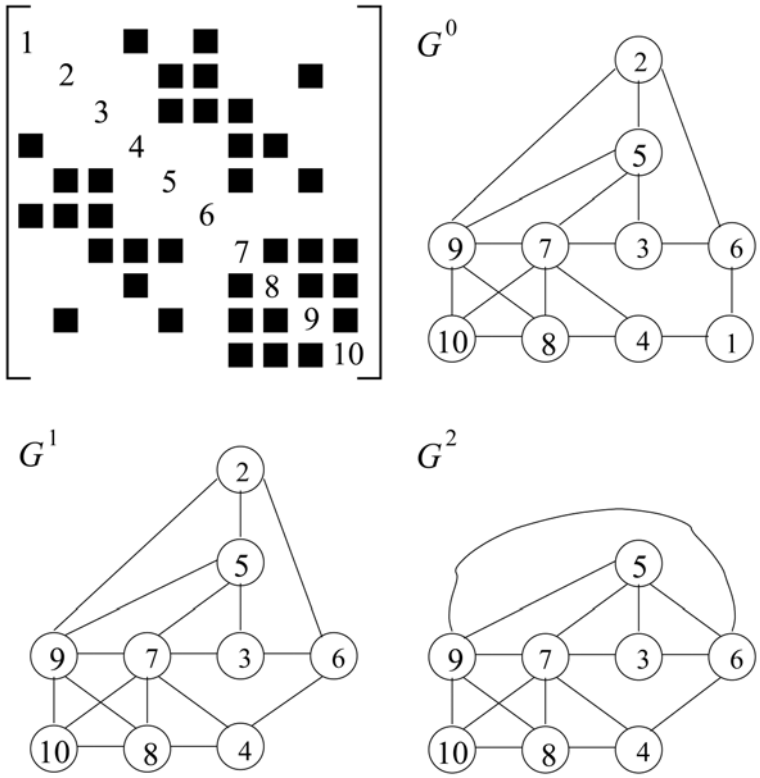

Fig. 2. Example of matrix and its elimination graph (initial and after first two steps)

At the beginning of the algorithm, an initial graph G^0 is constructed according to the non-zero structure of the matrix to be reordered. This graph is then manipulated by a sequence of steps; in each step one vertex with the minimum degree is removed until the graph is empty. When a vertex is removed from the graph, its adjacent edges are also removed, and new edges are added to the graph so the vertices originally adjacent to the removed vertex are all connected to each other. Removing a vertex with the minimum degree causes minimum addition of new edges and also minimum fill-in in

numerical factorization. In each step k, the graph G^k represents the non-zero structure of the partially factorized matrix. An example of a matrix and its elimination graph for the first two elimination steps is presented in *Fig. 2*, the complete figure with explanation can be found in [4].

2.2. Quotient graph

The addition of new edges that represent the fill-in to the elimination graph means the storage requirements grow unpredictably during the symbolic elimination process. This has been resolved by using a *quotient* graph [4], [5], whose storage requirements never exceed the size of the initial graph G^0.

The quotient graph is however more complicated and makes the computation of degrees somewhat difficult. To get the degree in the elimination graph, one only counts one set of edges (i.e. edges adjacent to one vertex); but in the quotient graph, one must merge and count several sets of edges (avoiding duplicates), which makes it the most computationally intensive part of the algorithm. There are several ways to cope with this problem.

2.3. Mass elimination

The number of degree computations in the quotient graph can be reduced considerably by the use of *super-variables* [4], [5]. A super-variable is a special vertex, which represents several regular vertices at the same time. Vertices represented by the super-variable are removed from the graph completely, greatly reducing the number of vertices and edges in the graph. Regular vertices can be seen as super-variables representing only one vertex.

Super-variables are created from vertices that form a *clique* - a fully connected subgraph, i.e. all vertices in a clique are interconnected. These vertices - also called *indistinguishable variables* - are equivalent in terms of their edge sets and any of them can be selected as the next pivot, because they have the same degree.

Vertices represented by a super-variable are eliminated together when the super-variable is selected as a pivot, saving several elimination steps (hence the term *mass elimination*). The extra time taken by the detection of indistinguishable variables is negligible in comparison to the time saved by the reduction in degree computations.

2.4. Approximate degrees

The complexity of degree computations in the quotient graph can be reduced by the use of *approximate degrees*. A degree can be approximated by its upper bound that is simpler and faster to compute than the exact degree. However, the ordering obtained by the approximate degrees is generally not as good as the one obtained by the exact degrees.

There are several methods of computing the approximate degree [4], [6], three of them were tested and are assessed in this paper.

3. Numerical tests

Numerical tests were performed using finite element solver PMD (Package for Machine Design) [7] written in FORTRAN 77. The solver uses minimum degree ordering and sparse matrix storage scheme to allow efficient solution of large finite element problems.

The objectives of these tests were to verify the effects of the degree computation methods and the use of mass elimination on both the ordering speed and quality and to determine the best configuration for solving real-world finite element problems. This is important because the performance of the minimum degree algorithm is heavily dependent on the actual implementation.

The tests were performed on various finite element problems ranging from small (approx. 10^2 equations) to large (approx. 10^6 equations).

3.1. Tested minimum degree algorithm configurations

Eight configurations of the minimum degree ordering were tested: four methods of degree computation both with and without mass elimination. Presented minimum degree algorithm implementation uses a quotient graph exclusively, because an implementation with the elimination graph would be too inefficient.

Algorithm configurations - degree computation methods:

- D1. Exact [3], [4], [5], it should yield the best but slowest ordering;
- D2. Approximate, proposed by Amestoy, Davis and Duff [4], it should yield the best or almost the best ordering while faster than D1;
- D3. Approximate, proposed by Ashcraft and Eisenstat [4]. A combination of D1 and D4, it should yield better ordering than D4 while still faster than D1;
- D4. Approximate, proposed by Gilbert, Moler and Schreiber [4], [6]. It is the simplest and fastest approximation.

3.2. Ordering time

The speed of the implemented minimum degree ordering algorithms measured by the ordering time is presented in *Fig. 3*. Algorithm configuration (D1 to D4) is on the *x*-axis and ordering time (scaled to the reference configuration) is on the *y*-axis. The reference configuration is D1, i.e. the exact degree computation with the mass elimination. The lines, which connect the points belonging to the same problem have no real meaning but to improve readability. The results obtained without the mass elimination are denoted by dashed lines and asterisks.

With the mass elimination (solid lines) the ordering speed generally increases considerably. The difference in ordering times between the configurations is about 60%. Surprisingly, algorithm D1 seems to be the fastest - its speed can only be matched by algorithm D4, which is however expected to yield much worse ordering.

Without the mass elimination (dashed lines) there is much larger difference in ordering speed between configurations. Algorithm D4 indeed proved to be the fastest. Algorithm D2 seems to be much faster than both algorithms D1 and D3, as predicted in

literature. Algorithm D3 is sometimes slower than D1, which is also somewhat surprising.

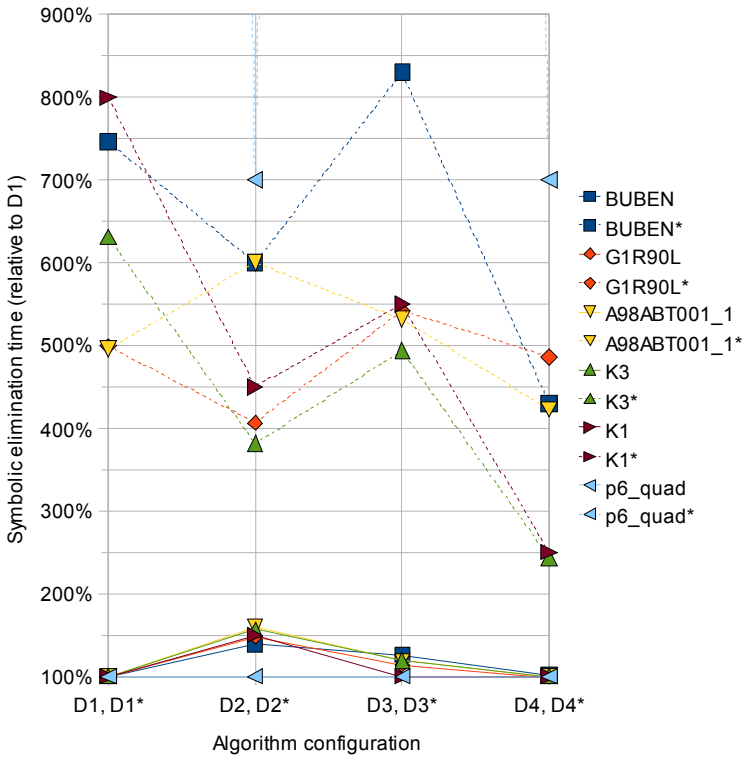

Fig. 3. Minimum degree algorithm ordering times (selected results)

Probable reason for the outstanding performance of algorithm D1 (exact degree computation) is the implementation of the minimum degree ordering, which was highly optimized at the programming level to minimize the number of mathematic operations performed on edge sets. Although all assessed algorithms use the same implementation, approximate degrees (especially algorithm D2) need some extra computation, which may eventually turn slower than 'plain' algorithm D1, in particular when mass elimination is enabled.

3.3. Ordering quality

The quality of the minimum degree ordering algorithms measured by the number of non-zeros in the factorized matrix is presented in *Fig. 4*. Algorithm configuration (D1 to D4) is on the *x*-axis and number of non-zeros after factorization (scaled to the reference configuration) is on the *y*-axis. The reference configuration is D1, i.e. the exact degree computation with the mass elimination enabled. As in the previous graph, connecting

lines were added for better readability. The results obtained without mass elimination are denoted by dashed lines and asterisks.

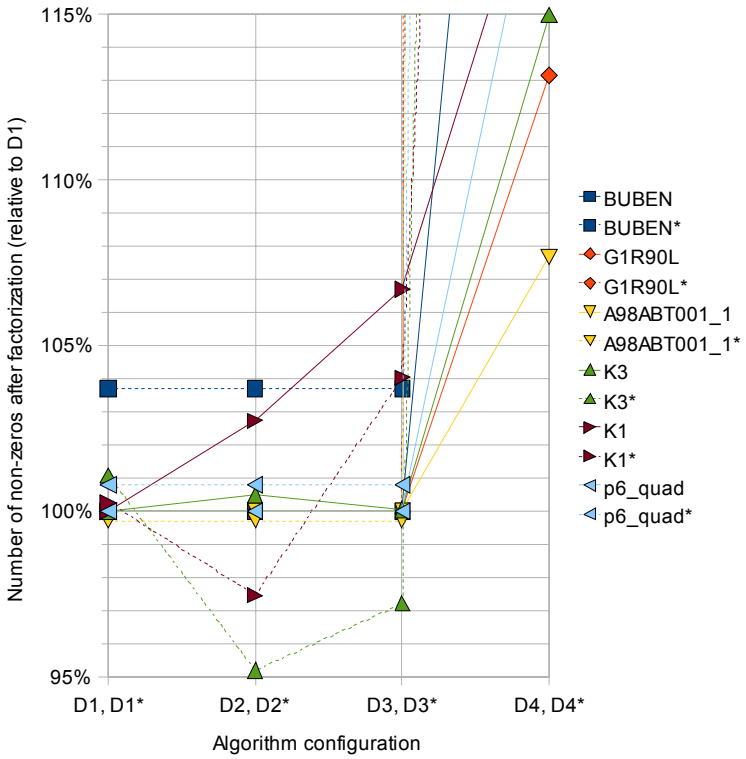

Fig. 4. Minimum degree algorithm ordering quality (selected results)

With the mass elimination (solid lines) the difference in ordering quality between algorithms D1, D2 and D3 is very small, while algorithm D4 yields considerably worse results.

Without the mass elimination (dashed lines), algorithms D1 to D3 yield only about 5% worse orderings than with the mass elimination, while algorithm D4 yields ordering worse by several hundreds of percent.

4. Conclusions

Several minimum degrees ordering algorithm configurations were tested on a set of finite element problems and their performance was compared and assessed.

Mass elimination was confirmed to be a very important feature because it not only considerably speeded up the minimum degree-ordering algorithm but also in most cases slightly improved the ordering quality. This is because eliminating indistinguishable variables together causes fewer fill-ins than eliminating the variables with the same

degree in a random order as in the original minimum degree algorithm. Therefore, only results with the mass elimination will be considered for concluding assessments.

Approximate degree computation proposed by Gilbert, Moler and Schreiber (configuration D4) was confirmed the fastest, but the ordering quality is considerably worse in comparison to the other configurations and is therefore not recommended to use for matrix reordering prior to factorization when low fill-in is needed.

Approximate degree computation proposed by Ashcraft and Eisenstat (configuration D3) usually gives relatively good orderings, however it is somewhat slow, even slower than exact degree computation (configuration D1).

Approximate degree computation proposed by Amestoy, Davis and Duff (configuration D2) was confirmed faster and in few cases even better than the exact degree computation. However, considering the mass elimination, assessed implementation of the exact degree computation (configuration D1) was found slightly faster.

In terms of performance, assessed implementation of the original minimum degree algorithm with quotient graph, mass elimination and exact degrees seems to outperform minimum degree algorithms with approximate degrees. This may, of course, be affected by the way the algorithms were implemented, thus, further thorough testing is needed to confirm these results.

Acknowledgements

This work was supported by the Czech Science Foundation project no. 101/09/1630 under AV0Z20760514.

References

[1] Ueberhuber C. W. *Numerical computation*, Springer, Berlin, 1994.

[2] Okrouhlík M., Höschl C., Nadrchal J., Plešek J., Pták S.: *Mechanics of non-rigid bodies, numerical mathematics and supercomputers* (in Czech), Institute of Thermo-mechanics ASCR, Prague, 1997.

[3] Tinney W. F., Walker J. W. Direct solutions of sparse network equations by optimally ordered triangular factorization. *Proc. of the IEEE*, Vol. 55, 1967, pp. 1801–1809.

[4] Amestoy P. R., Davis T. A., Duff I. S. An approximate minimum degree ordering algorithm, *SIAM J. Matrix Analysis & Application*, Vol. 17, No. 4, 1996, pp. 886–905.

[5] George A., Liu J. W. H. The evolution of the minimum degree-ordering algorithm, *SIAM Review*, Vol. 31, 1989, pp. 1–19.

[6] Gilbert J. R., Moler C., Schreiber R. Sparse matrices in MATLAB: design and implementation, *SIAM J. Matrix Analysis & Application*, Vol. 13, 1992, pp. 333–356.

[7] VAMET/Institute of Thermo-mechanics ASCR, PMD version f77.9, 2003.