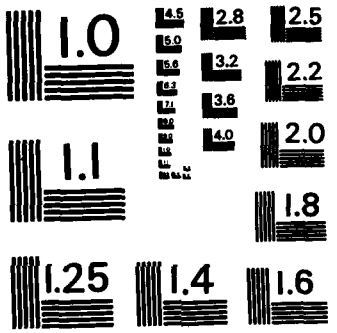
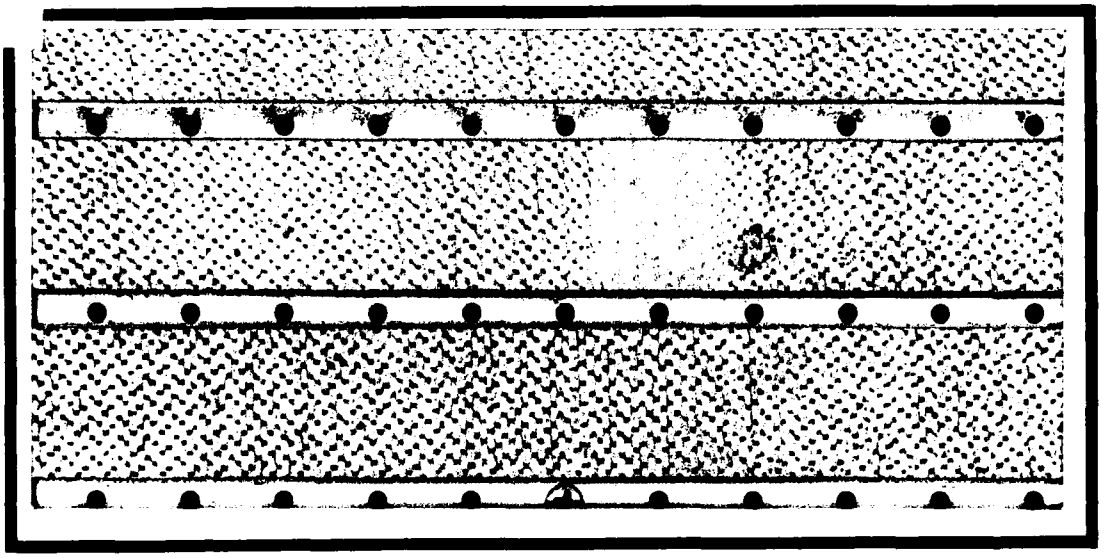
AD-A159 769 A MODIFIED FRONTAL TECHNIQUE SUITABLE FOR PARALLEL SYSTEMS(U) PITTSBURGH UNIV PA INST FOR COMPUTATIONAL MATHEMATICS AND APPLICATIONS R G MELHEM JUL 85 1/1

UNCLASSIFIED ICMA-85-84 N00014-85-K-0339 F/G 12/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

INSTITUTE FOR COMPUTATIONAL MATHEMATICS AND APPLICATIONS



Department of Mathematics and Statistics

University of Pittsburgh

Technical Report ICMA-85-84 July 1985

A MODIFIED FRONTAL TECHNIQUE SUITABLE FOR PARALLEL SYSTEMS*)

by

Rami G. Melhem**)

Institute for Computational Mathematics and Applications
Department of Mathematics and Statistics
University of Pittsburgh
Pittsburgh, PA 15260

*) This work was in part supported under ONR Contract N00014-85-K-0339.

**) On leave from the Department of Computer Science, Purdue University, West Lafayette, IN 47907.

(A-)

A MODIFIED FRONTAL TECHNIQUE SUITABLE FOR PARALLEL SYSTEMS

Rami G. Melhem

Department of Mathematics and Statistics

University of Pittsburgh

Pittsburgh, PA 15260.

ABSTRACT

Frontal techniques offer the potential for processing the assembly and the factorization phases of finite element analysis in parallel systems. However, the rows of the stiffness matrix are assembled and factored in different orders, thus depriving frontal solvers of the uniformity desired in parallel processing. On the other hand, band solution techniques handle the factorization phase in a very uniform way but do not interleave assembly and factorization. In this paper, we suggest a technique that borrows from both frontal and band solvers those characteristics that are advantageous for parallel processing. Moreover, book keeping and data manipulation are simpler in the suggested technique than in the classical frontal method. This makes the suggested technique also attractive for sequential systems.

KEY WORDS : Finite Elements - Frontal Methods - Numbering Schemes - Parallel Processing

ABBREVIATED TITLE : An Order Preserving Frontal Technique.

1. INTRODUCTION

The frontal solution technique [5] is a very effective means for reducing the computer time and the storage requirement for finite element analysis. Its central concept is the alternation between the assembly of the stiffness matrix H and its factorization. In order to be more specific, let the elements in the finite element grid be labeled by unique integers 1,...,m, and processed in that order. That is the corresponding element matrices $H^1, \cdots, H^m$ are accumulated in the global matrix H in the given order. Also let the nodes in the grid be numbered by the integers $1, \cdots, n$. If d is the degree of freedom at each node, then we may associate node i with the rows $(i-1)d+1, \cdots, id$ of H. In this paper, we will simplify the discussion by assuming that $d = 1$. However, it is easy to see that the results apply to the case $d \neq 1$ as well.

After the processing of an element e (the accumulation of H^e into H) and before the processing of element e+1, we may define the following two sets of rows of H

1) The set of partially assembled rows $\{i \mid i \in 1 \cup \cdots \cup e$ AND $i \in e+1 \cup \cdots \cup m\}$, where $i \in e$ denotes that i is a node in element e.

2) The set of ready rows $\{i \mid i \in e$ AND $i \notin e+1 \cup \cdots \cup m\}$. Any row in this set will not be modified by the processing of future elements. The union of the above two sets is called the active front at element e and is denoted by $F_a(e)$.

A frontal solver identifies, after the processing of each element e, the rows in $F_a(e)$ that are ready and uses each ready row i of H to eliminate the sub-diagonal elements in column i and then removes the ready rows from core memory. Hence, if $|F_a(e)|$ is the cardinality of $F_a(e)$, and $\bar{F} = \max\{|F_a(e)| : e=1,\cdots,m\}$, then the frontal solver needs to provide core storage for only $\bar{F}$ rows of H.

For large problems, frontal solvers have two advantages over band solvers, namely 1) they interleave the assembly and factorization of H, and hence eliminate the need to store H in secondary storage during the assembly and then to retrieve it during the factorization, 2) They require less core memory than band solvers because $\bar{F}$ is usually smaller than the bandwidth of the matrix H [2,4]. However, $F_a(e)$ consists of non-contiguous rows of H, which requires some indexing to keep track of the location of each row in memory. Also, some preprocessing is needed in order to determine the instant at which each row becomes ready (completely assembled).

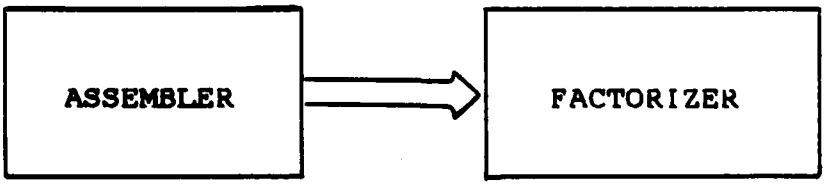


Figure 1

Clearly, the rows of H do not become ready in sequential order, which is a serious problem if the assembly and factorization are to be executed in parallel on different

hardware units (Fig 1). For example if an array processor is used for the factorization, then indexing becomes a source of inefficiency. The same applies if different processors are used for the assembly and the factorization in a multiprocessor system. Moreover all the special purpose hardware that are suggested in the literature for matrix factorization expect to receive the rows of H in a sequential order (see for e.g. [6] and [7]).

In this paper, we suggest a variation of the frontal technique that does release the ready rows of H in order. This variation has the added advantage that the instant at which each row is to be released to the factorizer is uniquely determined by a parameter δ, that is no preprocessing is needed to generate information about the instant at which each row becomes ready.

The size of the core memory needed by the assembler is proportional to the value of the parameter δ, which, then, has to be chosen in an optimal way. In Section 3, we describe an algorithm for the determination of δ_{min}, the optimal δ for a given problem, then, in Section 4, we restrict our attention to the sub-class of finite element grids that are commonly used in practical applications, and we derive an upper bound on δ_{min} for this sub-class.

2. An order preserving frontal technique

In the rest of this paper we will not distinguish between the application of the frontal technique to conventional or parallel architectures. More specifically, we will use the term "a row is consumed" to indicate that the row is factorized 'in core' in conventional computers, or that the row is passed to the factorization unit, in the case of parallel processing.

Let $\lambda(e) = \max\{i \mid i \in F_a(e)\}$. That is, immediately after the assembly of element e, $\lambda(e)$ is the row with the largest index in the active front. Also, let δ be an integer such that, after the precessing of any element e, rows $1,\cdots,\lambda(e) - \delta$ are ready (completely assembled), and hence, may be consumed.

The basic idea in our modified frontal technique is to find the minimum value of the integer δ. This value is called the width of the delayed front and is denoted by δ_{min}. Given δ_{min}, the assembly of H and its factorization may be interleaved as described by the following algorithm:

```
Last-consumed := 0 ;
For elements e=1, ... , m do
    [] Assemble H^e into H and determine λ(e) ;
    [] If e < m,
        Then consume rows Last-consumed , ... , λ(e)-δ_min
        Else consume rows Last-consumed , ... , n ;
    [] Last-consumed := λ(e)-δ_min ;
```

Clearly, after any iteration e, rows $\lambda(e)-\delta_{min}+1,\cdots,\lambda(e)$ of H are not consumed and have to be stored in memory. A circular buffer of size δ_{min} may be used to store these rows in sequential order. Here, note that even if a row ρ, $\lambda(e)-\delta_{min} < \rho \leq \lambda(e)$, is ready after the processing of element e, its consumption is delayed until after the processing of an element $\bar{e}$ with $\lambda(\bar{e})-\delta_{min} \geq \rho$. This is different from the classical frontal technique where rows are consumed as soon as they are ready.

The purpose of delaying the consumption of the ready rows of H is two folds: 1) to ensure that the rows of H are consumed in sequential order and 2) to allow for an automatic determination of the instant at which each row is to be consumed. The price to be paid is a larger memory requirement. However, with today's technology, this price is affordable as long as reasonable bounds may be imposed on the width of the delayed front δ_{min}. Such bounds will be discussed in Section 4.

In any frontal technique, the order at which the elements are processed is crucial because it determines the size of the active front. Hence, an element numbering is first chosen to minimize the active front, then the nodes are numbered according to their occurrences in the elements. More specifically, given an element numbering, the following algorithm is usually used to number the nodes:

ALG1

last-number := 0 ;

For elements e = 1,···,m Do

1) For each node v in e that is not numbered yet Do

1.1) last-number := last-number + 1 ;

1.2) Give v the number last-number ;

This type of two phase node numbering scheme has been studied in [3] where it is shown that if the elements are numbered using the reverse Cuthill-Mckee algorithm [1] then the profile, bandwidth and anticipated fill-in of the matrix H resulting from the two phase node numbering are comparable to those resulting from the best known heuristic node numbering scheme, namely the reverse Cuthill-Mckee algorithm.

If ALG1 is used to number the nodes in the grid, then the width of the delayed front δ_{min} may be easily determined provided that the element numbering is proper in the sense of the following definitions:

Definition 1: Given a specific numbering of the elements, an element e, $1 \leq e \leq m$ is called "wrapped" if any node in e is also in one of the previous elements 1,···,e-1.

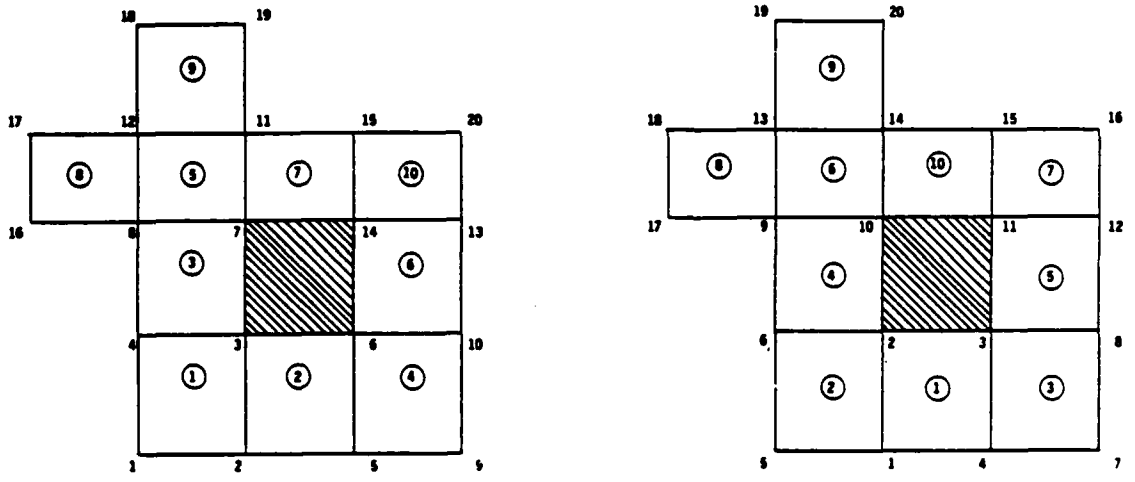
Definition 2: An element numbering is called "proper" if it does not result in any wrapped element. That is, it satisfies the property that any element e, $1 \leq e \leq m$, contains at least one node that is not in elements 1,···,e-1.

Proposition 1: Given a finite element grid and a proper element numbering, let the nodes in the grid be numbered by ALG1, and let the half-bandwidth of the resulting matrix H be $B_h = \max_{e=1,..,m} \{|g(e)-s(e)|\}$, where g(e) and s(e) are the largest and smallest node numbers, respectively, in element e. If each element contains at most k nodes, then

$$B_h - k < \delta_{min} \leq B_h$$

Proof: Consider the situation after the processing of any element e. From the proper numbering, any element r, r > e, contains at least one node not in elements $1,\cdots,e$. Hence, $g(r) > g(e)$, where g(r) and g(e) are the largest node numbers in elements r and e, respectively. But, from the definition of B_h, the smallest node number in element r satisfies $s(r) \geq g(r)-B_h > g(e)-B_h$. That is rows $1, \cdots, g(e)-B_h$ are not affected by the assembly of element r. Noting that this is valid for any r > e and that $\lambda(e) = g(e)$ in a proper labeling, we conclude that $\delta_{min} \leq B_h$.

Next, let $\bar{e}$ be the specific element that satisfies $B_h = g(\bar{e}) - s(\bar{e})$, and let $e = \bar{e}-1$. Clearly, at most k-1 nodes may be numbered in element $\bar{e}$, that is $g(\bar{e}) \leq g(e)+k$. Now, after the processing of element e, rows $1,\cdots,g(e) - (B_h - k)$ are not completely assembled because element $\bar{e} > e$ contains a node $s(\bar{e}) = g(\bar{e}) - B_h \leq g(e) + k - B_h$. Hence, row $s(\bar{e})$ will be affected by the assembly of element $\bar{e}$, which proves that $\delta_{min} > B_h - k$ ■



(a) Proper labeling (b) Non-proper labeling

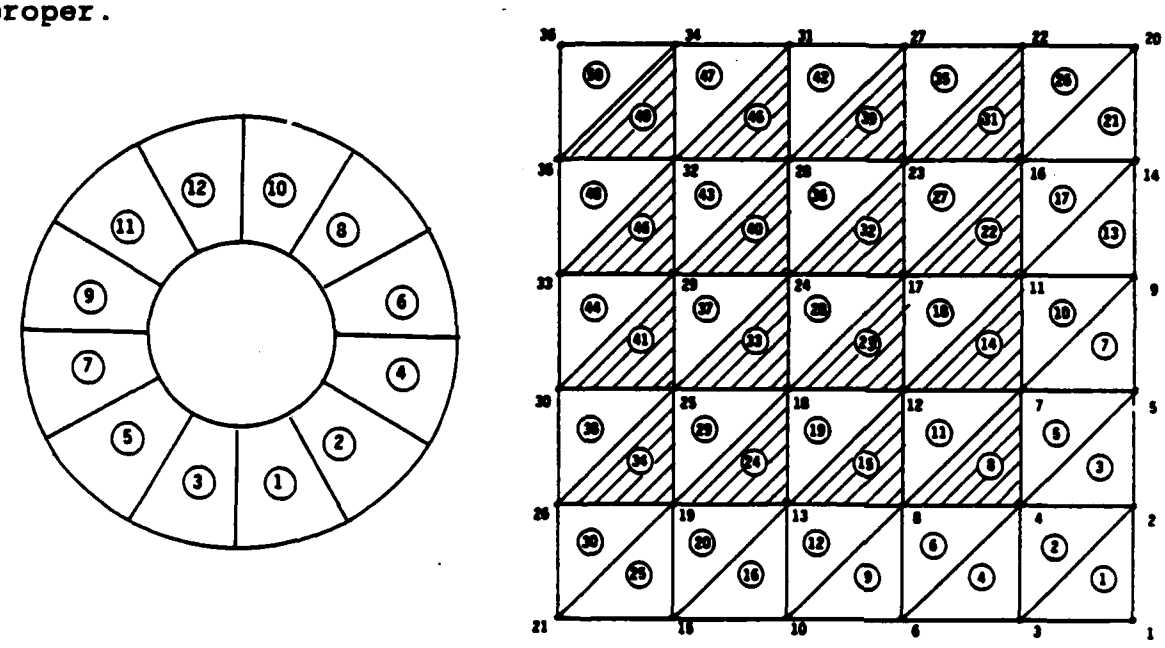
Figure 2

Proposition 1 states that if we chose $\delta = B_h$, then we will be away from the optimal δ_{min} by at most k. In Figure 2.a and 2.b, we give a proper and a non-proper element numbering, respectively, for the same grid (element numbers are enclosed in circles). The node numbering resulting from ALG1 is also shown. In both cases $B_h = 9$, but in the non proper case, after the assembly of element 9, row $g(9)-B_h=20-9=11$ is not completely assembled and will be affected by the assembly of element 10. In other words, $\delta_{min} > B_h$, which proves that proper element labeling is essential for the result of Proposition 1 to hold.

If the elements in the grid are of the Lagrangian type with $k > 4$, then each element contains at least one center node, and hence, any element labeling is proper. However, general conditions for the existence of proper labeling are hard to obtain. In Figure 3, we show two grids for which no proper element labeling exist. The choice of δ in such cases is discussed in the next section.

3. Chosing δ for general element labeling

Consider the triangular grid of Fig 3.b. Clearly, the elements that are hashed in the figure are wrapped, and hence the given element labeling is not proper. The corresponding node numbering (shown also in the figure) yields $B_h = 7$. For this numbering, row 16 is partially summed after the assembly of element 30, and hence $\delta_{min} \geq g(30)-16 = 26-16 = 10$. That is, some criteria other than $\delta = B_h$ should be used if the element labeling is not proper.



(a) (b)

Figure 3 - Grids that do not have proper labeling

The method that we suggest for the choice of δ is based on the idea of augmenting the given grid with dummy nodes so that no elements are wrapped. Each dummy node is given the same number as the last numbered node and the band width, B_a, corresponding to the augmented grid is computed. Then δ is taken to be equal to B_a. For example, applying this

procedure to the grid of Fig 3.b gives the grid of Fig 4, from which the augmented band-width is found to be 10.

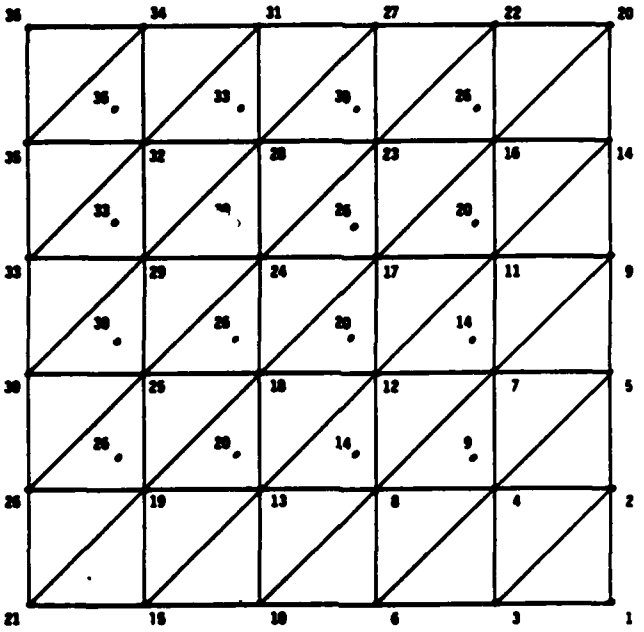


Figure 4 - The grid of Fig 3.a augmented with dummy nodes

Note that the augmented grid does not have to be constructed in order to compute B_a. It suffices to keep track, while numbering the nodes in ALG1, of the last node that has been numbered. More specifically, we may modify ALG1 to compute B_a as follows:

ALG2

last-number := 0 ; B_a = 0 ;

For elements e = 1,···,m Do

1) For each node v in e that is not numbered yet Do

1.1) last-number := last-number + 1 ;

1.2) Give v the number last-number ;

2) g(e) := last-number ;

Find s(e) ; the smallest node number in e ;

If (B_a < g(e) - s(e)) Then B_a := g(e) - s(e) ;

It is easy to check that if the element numbering is proper, then B_a computed in ALG2 reduces to B_h, the half-bandwidth of the matrix H.

Proposition 2: For a general element numbering, if $B_a > B_h$ then $\delta_{min} = B_a + 1$, else, if $B_a = B_h$, then

$$B_a - k < \delta_{min} \leq B_a + 1$$

where k is the number of nodes in each element.

Proof: Consider the situation after the assembly of a particular element e. For any element $r > e$, ALG2 implies that any node number ν in r satisfies $\nu \geq g(e) - B_a$, where g(e) is the largest node number in e. This is valid for any $r > e$, and hence rows $1,\cdots,g(e)-B_a-1$ are completely assembled and $\delta_{min} \leq B_a + 1$.

Now, for $B_a = B_h$ the lower bound is proved as in Proposition 1. This bound may be tightened if $B_a > B_h$, because this implies that there exists an element $\ddot{e}$ such that a dummy node, say μ, was added to $\bar{e}$ and $B_a = \mu - s(\ddot{e})$. But from ALG2, there exists an element $e < \ddot{e}$ such that $g(e) = \mu$. Hence, after the processing of element e, rows $1,\cdots,g(e)-B_a$ are not completely assembled, because row $s(\ddot{e})=g(e) - B_a$ will be modified during the assembly of element $\ddot{e}$. This implies that $\delta_{min} > B_a$ ■

Proposition 2 shows that the choice of $\delta = B_a+1$ is optimal if $B_a > B_h$ and is away from the optimal by at most k

if $B_a = B_h$.

The next question to be asked is: How large can B_a be compared to B_h?. This is important because it determines the maximum storage needed by the assembler. However, it seems impossible to obtain any bound on B_a if complete freedom is allowed in the construction of the grid and in the numbering of its elements. For this reason, we define in the next section the class of W-proper element numbering that excludes arbitrary strange grids and numberings.

4. Upper bounds on B_a for W-proper element numbering

The results reported by Law and Fenves [3] suggest the use of the Cuthill-Mckee (CM) algorithm or its reverse for numbering the elements in two phase node numbering schemes. In their paper, the following definition of adjacency is used:

Definition 3: Two elements in a finite element grid are called adjacent if they share a common edge.

With this definition, the CM algorithm may be described as follows:

ALG3 - The Cuthill-Mckee algorithm

```
i = 0 ; Level[0] = { a specific starting element } ;
Repeat until all elements are numbered
    i = i + 1 ;
    Consider the elements in Level[i-1] in order of
    ascending numbering. For each element e, determine
    the elements that are adjacent to e, number them and
    include them in Level[i]
```

The specification of the scheme used to number the elements does not exclude grids with arbitrary strange shapes. The following definition imposes some regularity on both the grid topology and the numbering scheme:

Definition 4: An element numbering is called W-proper, if each wrapped element $\bar{e}$ shares a node ν with an adjacent element e, such that ν is not in elements $1,\cdots,e-1$. Note that this

implies that $e < \bar{e}$ and that e is not wrapped.

More descriptively, if ALG1 is used to number the nodes in the grid, then each element $\bar{e}$ that does not contain a new node should contain at least one node ν that have been numbered in an adjacent element e. For example, the numberings in Figure 3 are W-proper, while the CM numbering of the grid shown in Figure 5 is not W-proper. More specifically, in Fig 5, element 25 is wrapped and its only adjacent element, namely 24, is also wrapped. For this example, it is easy to see that B_h=10 and B_a=22.

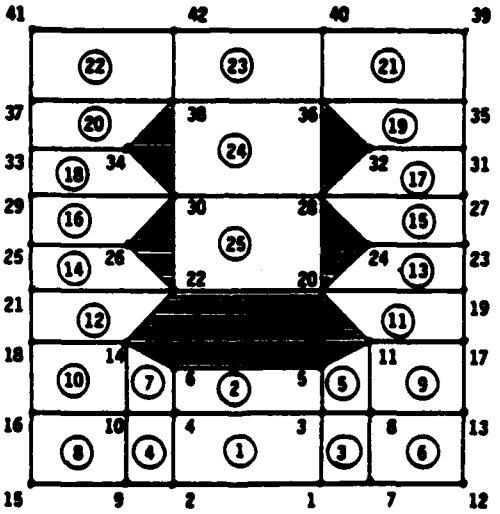


Figure 5 - A non W-proper element numbering

As is clear from the above example, non W-proper numbering may be obtained only on very strange grids. Moreover, the following may be proved:

Proposition 3: If the elements in the grid are of the serendipity type, that is contain nodes on the edges of the elements in addition to those on the corners, then any element numbering of the grid is W-proper.

Proof: Let e and $\bar{e}$ be two adjacent elements, and let $\bar{\bar{e}} > e$. From the hypothesis of the proposition, there is a node ν on the common edge of e and $\bar{e}$ so that ν cannot be in any other element in the grid. Hence ν is not in elements $1,\cdots,e-1$ and hence e cannot be wrapped ■

Now, we are ready to establish a bound on B_a.

Proposition 4: If the CM algorithm is used to number the elements of a grid, and the resulting numbering is W-proper, then

$$B_a \leq 2\, B_h$$

Proof: Let $\bar{e}$ be the element that satisfies

$$B_a = \mu - s(\bar{e}) = g(\bar{u}) - s(\bar{e}) \qquad (1)$$

where μ is the dummy node added to $\bar{e}$ and $\bar{u} < \bar{e}$ is the first element before $\bar{e}$ that is not wrapped.

From the hypothesis, there is an element e adjacent to $\bar{e}$ such that $e < \bar{e}$, and there exists a node ν in both e and $\bar{e}$ such that ν is not in elements $1,\cdots,e-1$. By the definition of B_h,

$$\nu - s(\bar{e}) \leq B_h \qquad (2)$$

Given that Level[0] in CM algorithm contains only one node, then $\bar{u}$ is in a level $i > 1$ and there exists an element u in level $i-1$ such that $\bar{u}$ is adjacent to u. In other words,

there is a node λ in both u and $\bar{u}$ that satisfies

$$g(\bar{u}) - \lambda \leq B_h \tag{3}$$

$$\lambda \leq g(u) \tag{4}$$

Moreover, $u \leq e$ because if $u > e$ then the CM algorithm would not number $\bar{u}$ before $\bar{e}$. Hence, any node not in elements $1,\cdots,e-1$ has a number larger than $g(u)$. In particular

$$\nu \geq g(u) \tag{5}$$

From (2), (3), (4) and (5) we get

$$g(\bar{u}) - s(\bar{e}) \leq 2\ B_h$$

The result then, follows directly from (1)■

5. Conclusion

We presented a method for interleaving the assembly and solution stages of the finite element analysis. This method differs from the classical frontal technique in the following:

1) The rows of the assembled matrix are made available for factorization in order. This is important for parallel processing.

2) The instant at which each row is made available is determined automatically, rather than through elaborate preprocessing.

3) The storage required by the assembler is determined by the width of the delayed front δ_{min} rather than the maximum size of the active front $\bar{F}$. Although δ_{min} is usually larger than $\bar{F}$, we proved that, for the type of meshes encountered in practical applications, $\delta_{min} \leq 2\, B_h < B$, where B is the bandwidth of the stiffness matrix H.

4) The rows of H are stored in order. Hence, no indexing is needed to keep track of the location of each row in memory.

In other words, features from both frontal and band solvers are combined in a method that is easy to implement on either parallel or uniprocessor systems.

Acknowledgement

I am pleased to thank Werner Rheinboldt and Kincho Law for helpful discussions and comments.

References

1. E. Cuthill and J. Mckee, ''Reducing the Bandwidth of Sparse Symmetric Matrices.,'' Proc. of the 24th National Conference of the ACM, pp.157-172 (1969).

2. I. S. Duff, ''Design Features of a Frontal Code for Solving Sparse Unsymmetric Linear Systems Out-of-core,'' SIAM J. on Scientific and Statistical Computing 5(2), pp.270-280 (1984).

3. S. J. Fenves and K. H. Law, ''A Two Step Appoach to Finite Element Ordering,'' Int. J. for Numerical Methods in Engineering (1983).

4. P. Hood, ''Frontal Solution Program for Unsymmetric Matrices,'' Int. J. for Numerical Methods in Engineering 10, pp.379-399 (1976).

5. B. Irons, ''A Frontal Solution Program for Finite Element Analysis,'' Int. J. for Numerical Methods in Engineering 2, pp.5-32 (1970).

6. K. H. Law, ''Systolic Arrays for Finite Element Analysis,'' Computers and Structures 20 (1985).

7. R. G. Melhem, ''On the Design of a Pipelined/Systolic Finite Element System,'' Computers and Structures 20, pp.67-75 (1985).

END

FILMED

11-85

DTIC