Setting up the cell neighbour array and the list of external faces for a finite volume mesh

C.G. du Toit[1]

(Received April 1997; Final version February 1998)

Three different schemes to sort the cell faces in a finite volume mesh into external and internal faces and to set up the cell neighbour array are compared. In each case a list of all the cell faces is systematically compiled whilst the faces of each cell are processed. As a face is added to the list, it is compared with the faces already in the list to determine whether it matches any of the faces. The faces can therefore also be sorted into external and internal faces at the same time. In the first algorithm each face in the list is checked until a face is found which matches the new face or the end of the list is reached. In the second algorithm a linked list of all the unmatched faces is maintained and the new face is therefore only compared with those in the linked list until a matching face is found or the end of the list is reached. In the third scheme the smallest nodal number associated with the new face is identified. Linked lists are compiled of all the unmatched faces which share the same smallest nodal number. When searching for a matching face, it is therefore only necessary to compare the new face with those in the same linked list.

Introduction

In a finite volume calculation a cell must know the cells by which it is surrounded. This information is of vital importance for the setting up of the system(s) of linear algebraic equations.[1] In the case of a topologically regular mesh indexing can be used to obtain this information. However, in the case of a topologically irregular mesh a cell neighbour array must be compiled to provide this information.

In comparison, in a typical finite element calculation it is not necessary that an element should know the elements by which it is surrounded. However, situations do arise where this information is also required in the finite element context. Du Toit[2] developed a finite element based Lagrangian tracking procedure to follow solid particles through a flow field. In this case it is necessary to know the neighbours of each element. This information is required to follow the particles when they pass from one element to the next as they traverse the flow field.

Also, before the refinement procedures can be invoked in a discontinuous h-adaptive finite volume or finite element scheme, the macro-edge or cell or element connectivity array which defines the neighbours of the macro or root cells or elements must be initialized.[3,4,5,6,7] Also in the case of quadtree or octree grid generation[8] it is necessary to identify the cells (or panels) which are located next to any cell (or panel) of interest. Once the neighbours of the initial cells are known, the changes in the cell connectivities can be recorded as the mesh is refined or unrefined.

During the setting up of a finite volume or finite element model the boundary conditions associated with solid walls which lie on the boundary of the computational domain can be defined automatically if the cell faces which coincide with these walls are known.[9]

The neighbours of the cells in a mesh can be determined by searching for the internal faces of the mesh and finding the cells which are associated with each of these faces. Whilst the faces which form the outer shell of the computational domain are those faces which are each only associated with one cell.[9]

This investigation is based on the approach followed by Du Toit *et al.*[10] In this paper three different algorithms, which can be used to sort the faces in a mesh into external and internal faces and to set up the cell neighbour array, are discussed and compared. The performance of the three schemes are evaluated with the aid of a number of examples.

Theoretical overview

Let $F_E = \{F_1, ..., F_{NE}\}$ be the set of all the external faces in the finite volume mesh with NE the number of external faces. Also let $F_I = \{F_{NE+1}, ..., F_{NF}\}$ be the set of all internal faces (an internal face is the interface between two adjacent cells) in the finite volume mesh with NF the total number of faces in the mesh. The set F_T of all faces in the mesh is therefore given by:

$$F_T = F_E \cup F_I \tag{1}$$

Further, let $f^i = \left\{f_1^i, ..., f_{nf}^i\right\}$ be the set of all the faces of the i-th cell with nf the number of faces in the cell (or cell face set). The set of internal faces F_I is then given by:

$$F_I = \bigcup_{i=1}^{NCL} \left[\bigcup_{j=1}^{NCL} f^i \cap f^j \right] \qquad \text{for } j \neq i \tag{2}$$

with NCL the number of cells in the mesh, whilst the set of external faces F_E is given by:

$$F_E = \bigcup_{i=1}^{NCL} \left[f_k^i \notin F_I \right] \qquad \text{for } k = 1, ..., nf \tag{3}$$

[1]School for Mechanical and Materials Engineering, Potchefstroom University for CHE, Private Bag X6001, Potchefstroom, 2520 South Africa

Eq. (2) implies that the set F_I is constructed by comparing the faces of each cell with the faces of every other cell in the mesh. The set F_E, according to eq. (3), is then constructed by comparing the faces of each cell with those in F_I and retaining the faces which are not elements of F_I. These procedures are very time consuming, particularly in the case of large grids.

A less time consuming procedure to construct the sets F_T, F_I and F_E is as follows. The set F_T is given by:

$$F_T = \bigcup_{i=1}^{NCL} [f_k^i \notin F_T^{i-1}] \qquad \text{for } k = 1, ..., nf \qquad (4)$$

where $F_T^{i-1} \subset F_T$ is the set of cell faces already added to F_T during the processing of cells 1 to $i-1$. Each time a cell face is added to the set (processed), it is only compared with those faces already in F_T. If $f_k^i \in F_T^{i-1}$, it means that f_k^i is an internal face (interface between two adjacent cells) and that it can be tagged. The set F_I is then given by all the tagged faces in F_T and F_E by all the other (untagged) faces in F_T. The sets F_I and F_E are, therefore, also systematically constructed along with F_T as the cell faces are processed. This, in essence, corresponds to the procedure described by Shaw.[9]

The algorithms discussed in the following sections are based on a slight adaptation of the latter approach. Instead of the set F_T, the set f_t, which contains all the individual cell faces of all the cells, is constructed

$$f_t = \bigcup_{i=1}^{NCL} f^i \qquad (5)$$

As in the case of eq. (4), each new cell face is compared with those faces already in f_t whilst the set is being compiled. If two faces are found to be coupled (identical, i.e. the same interface), both of them are tagged. The difference between the various algorithms, which will be illustrated, lies in the way in which the search for a coupled (identical) face through the list of faces already in the set f_t is conducted.

Although hexahedral cells are used throughout the paper, this is only for illustrative purposes, as the theory applies to any type of two-dimensional or three-dimensional cell. The only requirement is that there should be inter-cell continuity. Two adjacent cells must, therefore, share a full face.

Algorithms

Face record

The following data structures are assumed to be available:

- NODSEQ(INODF,JFACE) is the local node number of the INDOF-th node of the JFACE-th face of a cell (or the master cell). Consider the master hexahedral cell shown in Figure 1. The associated face topology array NODSEQ is given in Table 1. The columns give the (local) numbers of the nodes associated with the cell faces 1 to $nf = 6$. The fourth cell face, for example, is defined by the set of nodes f_4= {1,5,8,4}^{T}.
- KCELL(INODE,JCELL) is the global node number of the INODE-th node of cell JCELL. The cell topology array, for example, for the two cell mesh (model) shown in Figure 2 is given in Table 2. The second cell, for example, is defined by the set of global nodes c^2 = {5, 6, 8, 7, 9, 10, 12, 11}^{T}.

Table 1 Face topology array NODSEQ for master hexahedral cell

	JFACE					
INODF	1	2	3	4	5	6
1	2	3	5	1	1	1
2	3	4	6	5	2	4
3	7	8	7	8	6	3
4	6	7	8	4	5	2

Table 2 Cell topology array for two cell mesh

	JCELL	
INODE	1	2
1	1	5
2	2	6
3	4	8
4	3	7
5	5	9
6	6	10
7	8	12
8	7	11

The following additional data structures are needed:

- KFACS(IFACS,J) is the J-th field of the record of the IFACS-th face. An illustration of the layout of the face record array KFACS is shown in Table 3. N1 to N4 are the numbers of the global nodes associated with the face IFACS, with N1 the smallest nodal number and the rest in right-handed order. ICELL is the number of the cell (or cell face set f^i) of which the face IFACS is a member and JFACE is the local face number of the face IFACS in cell ICELL. JFACS is the number of the face which is identical (coupled) to the face IFACS. If JFACS = 0, the face is uncoupled (external).
- INEXT(IFACS) is the number of the face in the linked list of uncoupled faces to which the face IFACS points. For example, in Table 4 face 1 points to face 2, which again points to face 4, etc.

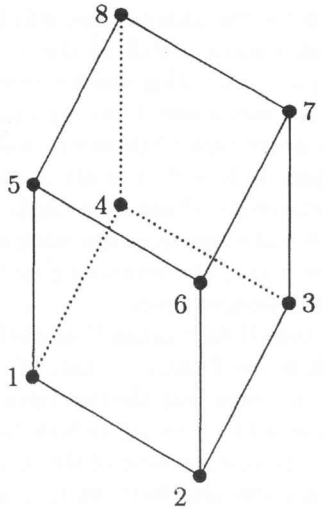

Figure1 Master hexahedral cell with local node numbers

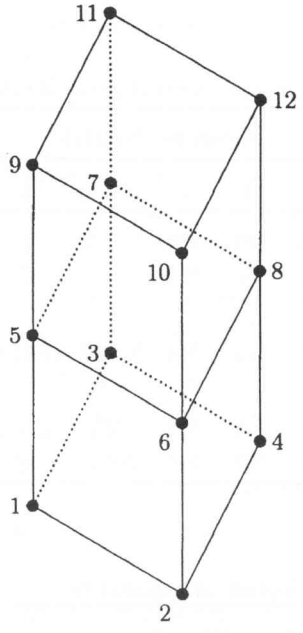

Figure 2 A mesh of two hexahedral cells

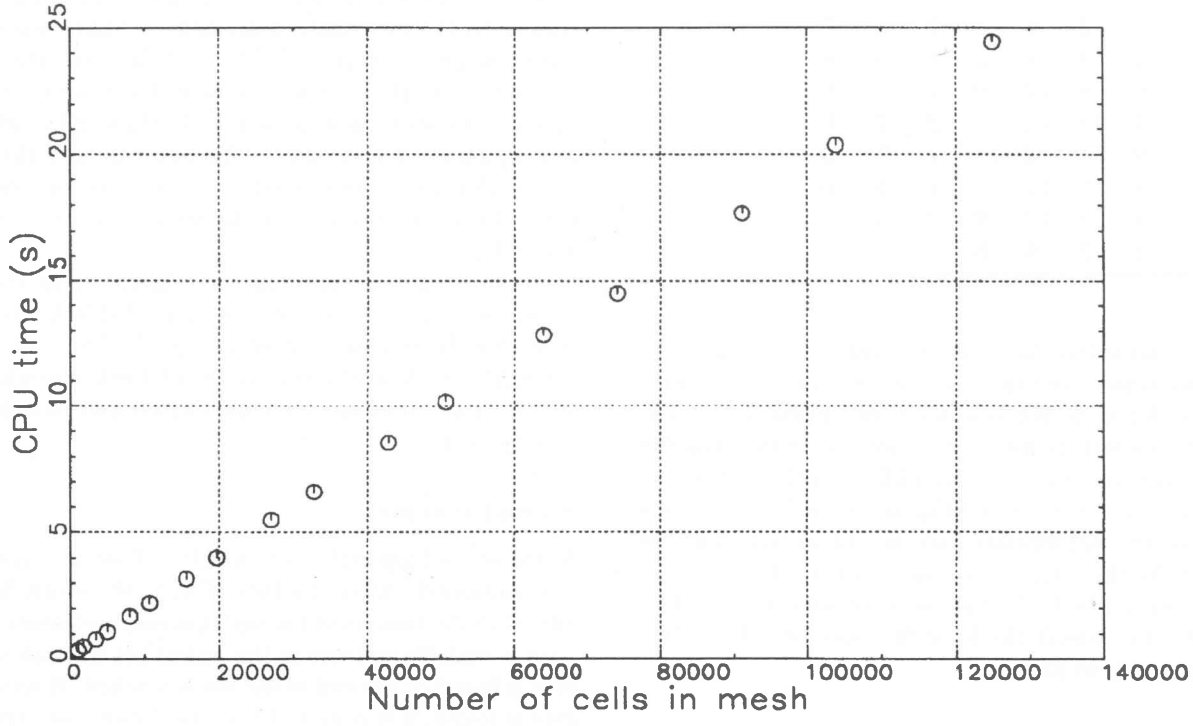

Figure 3 CPU time vs. number of cells for algorithm 3.

- KMINN(INODS) is the number of the first uncoupled face for which the global node INODS is the smallest nodal number. For example, in Table 5 the first uncoupled face for which the global node 5 is the smallest node, is face 10.

Table 3 Layout of face record array KFACS

Face	Fields of KFACS						
	1	2	3	4	5	6	7
1	N1	N2	N3	N4	1	1	JFACS
2	N1	N2	N	N4	1	2	JFACS
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
IFACS	N1	N2	N3	N4	ICELL	JFACE	JFACS
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
NFACS-1	N1	N2	N3	N4	*NCL*	*nf* − 1	JFACS
NFACS	N1	N2	N3	N4	*NCL*	*nf*	JFACS

Table 4 Result of algorithm 1

IFACS	Fields of KFACS							INEXT
	1	2	3	4	5	6	7	
1	2	4	8	6	1	1	0	2
2	3	7	8	4	1	2	0	4
3	5	6	8	7	1	3	12	0
4	1	5	7	3	1	4	0	5
5	1	2	6	5	1	5	0	6
6	1	3	4	2	1	6	0	7
7	6	8	12	10	2	1	0	8
8	7	11	12	8	2	2	0	9
9	9	10	12	11	2	3	0	10
10	5	9	11	7	2	4	0	11
11	5	6	10	9	2	5	0	0
12	5	7	8	6	2	6	3	0

As the individual faces of each cell (or cell face set) are being processed, the face records are added to the face record array KFACS. When a face is processed, the global node numbers which define the face are first of all extracted from the cell topology array KCELL with the aid of the cell face topology array NODSEQ and stored in the fields N1 to N4 of the appropriate face record in right-handed order, with Nl the smallest nodal number. The number of the cell with which the face is associated is stored in the field ICELL, whilst the local face number of the face is stored in the field JFACE.

Comparison of faces

Two typical neighbouring hexahedral cells are shown in Figure 2. In this case the internal face defined by the nodes 5, 6, 8, and 7 is shared by the cells and thus forms the interface between the two cells. It is normal practice to specify the nodes which define each face of a cell according to the right-hand rule such that the unit vector normal to each face points outwards. This means that the sequence in which the nodes for the shared face will be specified for the lower cell in Figure 1, will be the reverse of the sequence for the upper cell. This can be seen in Table 4 when the records for faces 3 and 12 are compared.

The following procedure is therefore followed when two faces are compared. It is first of all checked whether the faces are not members of the same cell. If not, the nodal numbers N1 for the two faces are compared. If they are found to be the same, the remaining nodal numbers are then compared in reverse order.

Let us assume that IFACS is the IFACE-th face of cell ICELL and JFACS is the JFACE-th face of cell JCELL. When the two faces are identical, the two faces are coupled by setting the field JCELL of the IFACS-th face record to JCELL and the corresponding field of the JFACS-th face record to ICELL. A particular face can be shared by two cells only. This can also be seen when the records for faces 3 and 12 in Table 4 are compared.

First scheme

In the first algorithm the face IFACS is added to the list and then compared with each uncoupled face before it in the list until an identical face is found or the end of the list is reached. Each face already in the list, therefore, is first of all checked to see whether it is coupled or uncoupled by looking at the field JCELL of the face record. This is very similar to the procedure described by Shaw,[9] except that Shaw assigns a unique label to each face and then checks for matching labels. After the faces have been sorted, the face record array is searched and a linked list of all the uncoupled faces is compiled. To facilitate this the linking vector INEXT, as described in the section Face record, is used. This, for example, can be seen in the last column of Table 4.

The result of applying this algorithm to the mesh, shown in Figure 2, is then given in Table 4. It can be seen that faces 3 and 12 are coupled. The first cell has a neighbour on side 3 and the second cell a neighbour on side 6. The other faces are uncoupled and linked via the linking vector.

Second scheme

In the second algorithm a linked list of all uncoupled faces is maintained. After the face IFACS, therefore, has been added to the face record array, it is only necessary to compare it with those faces in the linked list until an identical face is found or the end of the list is reached. If an identical face is found, it is removed from the linked list. Otherwise the face IFACS is added to the linked list. At the end of the sorting process the linked list of uncoupled faces is, therefore, immediately available.

Table 5 Result of algorithm 3

IFACS	Fields of KFACS							INEXT	INODS	KMINN
	1	2	3	4	5	6	7			
1	2	4	8	6	1	1	0	0	1	4
2	3	7	8	4	1	2	0	0	2	1
3	5	6	8	7	1	3	12	0	3	2
4	1	5	7	3	1	4	0	5	4	0
5	1	2	6	5	1	5	0	6	5	10
6	1	3	4	2	1	6	0	0	6	7
7	6	8	12	10	2	1	0	0	7	8
8	7	11	12	8	2	2	0	0	8	0
9	9	10	12	11	2	3	0	0	9	9
10	5	9	11	7	2	4	0	11	10	0
11	5	6	10	9	2	5	0	0	11	0
12	5	7	8	6	2	6	3	0	12	0

Table 6 Detail of example models and performance data for algorithms

Cells	Cell faces			CPU time (s)		
	Total	External	Internal	1	2	3
10 × 10 × 10	6000	600	5400	0.36	0.51	0.37
20 × 20 × 20	48000	2400	45600	109.29	11.08	1.71
30 × 30 × 30	162000	5400	156600	2501.80	127.20	5.69
40 × 40 × 40	384000	9600	374400	14714.97	556.51	13.33
200 × 2 × 20	48000	8880	39120	117.20	41.70	1.77
400 × 1 × 20	48000	16840	31160	127.38	79.93	1.78
8000 × 1 × 1	48000	32002	15998	141.27	144.85	1.79
400 × 20 × 1	48000	16840	31160	127.64	79.67	1.79
1 × 400 × 20	48000	16840	31160	127.92	80.42	1.78
20 × 400 × 1	48000	16840	31160	130.03	77.86	1.80
1 × 20 × 400	48000	16840	31160	130.13	78.07	1.78
20 × 1 × 400	48000	16840	31160	129.82	78.06	1.79

Third scheme

In the third algorithm separate linked lists are maintained of all the uncoupled faces which share the same smallest nodal number, i.e. have the same nodal number in the field N1 of the face record array KFACS. For this purpose the pointer vector KMINN, described in the section Face record, is used. KMINN(INODS) points to the first uncoupled face in the face record array KFACS for which INODS is the smallest nodal number. The linking vector INEXT is still used to provide the links to the other faces in the linked list. When the face IFACS, therefore, has been added to the face record array, it is only necessary to compare it with those faces in the link list associated with nodal number stored in the field N1 of the IFACS-th face record. If an identical face is found, it is removed from the linked list, otherwise the face IFACS is added to the list.

The result of applying this algorithm to the mesh shown in Figure 2, is given in Table 5. The linked lists of uncoupled faces can be seen, whilst the faces 3 and 12 are again coupled. Note that it is by chance that this model has the same number of faces and nodes. The list of uncoupled faces is not ordered and if an ordered list is required, the procedure employed in the first scheme to set up the linked list can be used to obtain the list.

In all three cases the information contained in the fields ICELL, JFACE and JFACS of each cell face record can be used directly to extract the numbers of the cells surrounding each cell, or to set up the cell neighbour array which will be similar to the cell topology array KCELL, except that it contains the numbers of the cells surrounding a particular cell.

Evaluation

Models

Twelve meshes have been set up to compare the performance of the algorithms discussed in the previous sections. The details of the meshes are presented in Table 6. Note the total number of cell faces, the number of external cell faces and the number of internal cell faces in each case. The models can be divided into two categories. In the first category the number of cells varies, whilst in the second category the number of cells is fixed. The second category, again, can also be divided into two subcategories. In the first the number of external and internal faces varies, whilst in the second the numbers are fixed.

The meshs $I \times J \times K$ have been generated by first incrementing in the I-direction, then in the J-direction and lastly in the K-direction.

Results

The computations were performed on a Aspen Durango workstation with an Alpha AXP 21164-433 MHz CPU and 512 Mb of RAM. The CPU times for the various cases are also tabulated in Table 6. These times include the time to read the cell data from file and to write the results to file. These times are also the result of a single computation in each case. A few spot checks have shown that these values are representative.

First of all, focusing on the first category, it can be seen that there is an almost exponential increase in the CPU times for the first two algorithms as the number of cells, and therefore the number of external and internal faces, increases. This is especially true of the first algorithm. However, in the case of the third algorithm the increase in CPU time is almost linear. The latter conclusion is confirmed by the results shown in Figure 3. The results obtained for two grids consisting of 512×10^3 and 10^6 cells, respectively, also confirmed the linear relationship. The larger the model becomes, the more marked the superior performance of the third algorithm becomes.

In the first subcategory of the second category the effect the variation – in the number of external and internal faces, for the same number of cells – has on the CPU times of the first and second algorithms can again be seen. The second algorithm, although more efficient than the first algorithm, seems however to be more sensitive to the variation in the number of external and internal faces. In the case of the third algorithm the variation in the number of external and internal faces seems to have no effect on the CPU times. In the second subcategory of the second category the order in which the grid is generated – and therefore the order in which the faces are processed, for the same number of cells, external faces, and internal faces – seems to have no effect on the CPU times for any of the algorithms.

It can therefore be concluded that the third algorithm is by far the most efficient algorithm of the three. The CPU time for the third algorithm also only increases linearly with the increase in the number of cells and seems to be almost unaffected by the variation in the number of external and internal faces for a fixed number of cells.

Conclusions

In this paper three different algorithms to sort the element faces in a finite volume mesh into external and internal faces and to set up the cell neighbour array were compared. In each case a list of all the cell faces is systematically compiled whilst the faces of each cell are processed. As a face is added to the list it is compared with the faces already in the list to determine whether it matches any of the faces. The faces are therefore also sorted into external and internal faces whilst the cell neighbour array is being set up.

In the first algorithm each face in the list is checked until a face is found which matches the new face or the end of the list is reached.

In the second algorithm a linked list of all the unmatched faces is maintained and the new face is therefore only compared with those in the linked list until a matching face is found or the end of the list is reached.

In the third scheme the smallest nodal number associated with the new face is identified. Linked lists are compiled of all the unmatched faces which share the same smallest nodal number. When searching for a matching face it is therefore only necessary to compare the new face with those in the same linked list.

Tests performed on a number of meshes have shown that the third algorithm is by far the most efficient algorithm of the three. The CPU time for the third algorithm increases linearly with the increase in the number of cells and it seems to be almost unaffected by the variation in the number of external and internal faces for a fixed number of cells.

References

1. SV Patankar. *Numerical heat transfer and fluid flow.* McGraw-Hill, 1980.

2. CG du Toit. Tracking particles through a flow field with obstructions. In: *Proceedings of Second National Symposium on Computational Fluid Dynamics.* Vereeniging, SA, 1991, 222–235.

3. JT Oden, T Strouboulis & P Devloo. Adaptive finite element methods for the analysis of inviscid compressible flow: Part 1: Fast refinement/unrefinement and moving mesh methods for unstructured meshes. *Computer Methods in Applied Mechanics and Engineering,* **59,** 1986, 327–362.

4. P Devloo, JT Oden & T Strouboulis. Implementation of an adaptive refinement technique for the SUPG algorithm. *Computer Methods in Applied Mechanics and Engineering,* **61,** 1987, 339–358.

5. P Devloo, JT Oden & P Pattani. An *h-p* adaptive finite element method for the numerical simulation of compressible flow. *Computer Methods in Applied Mechanics and Engineering,* **70,** 1988, 203–235.

6. L Demkowicz, JT Oden, W Rachowicz & O Hardy. Towards a univerisal *h-p* adaptive finite element strategy, Part 1: Constrained approximation and data structure. *Computer Methods in Applied Mechanics and Engineering,* **77**, 1989, 79–112.

7. KC Wang & GF Carey. Adaptive grids for coupled viscous flow and transport. *Computer Methods in Applied Mechanics and Engineering,* **82**, 1990, 365–383.

8. A Saalehi & AGL Borthwick. Quadtree and octree grid generation. *International Journal of Engineering,* **9,** 1996, 9–18.

9. CT Shaw. *Using computational fluid dynamics.* Prentice Hall, 1992.

10. CG Du Toit, AE Marquardt & LM Toerien. Finding Common Element Faces. In: *Proceedings of the llth Symposium on Finite Element Methods in South Africa.* Cape Town, SA, 1992, 143–154.